The Efficient Computing Hypothesis

Normative Theories of Neural Computation

William Dorrell

PhD Thesis – UCL – August 2025

A (Very) Solemn Declaration

I, William Dorrell, confirm that the work presented in my thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

An (Abstract) Abstract

The defining question of systems neuroscience is 'how does neural activity relate to computation?' Normative theories are one of many useful approaches to answering this question. They proceed by modelling neural activity as the solution to an optimisation problem. The paradigmatic example is the efficient coding hypothesis, in which neural activity is modelled as a faithful but efficient encoding of information. Efficient cod theory is widely successful, but mainly in sensory systems whose purpose seems well approximated as sending compact summaries of information to cortex. We argue that the rest of the brain is not just communicating information, it is computing with it, and our normative theories should reflect that. The other major normative approach, connectionism, aligns with this idea. It models neural activity using a task-optimised artificial neural network, and since these networks can perform most tasks, connectionism can be used to study almost any neural computations. However, the complexity of the networks sacrifices tractability: neurons in artificial networks are often as inscrutable as their biological cousins. Therefore, this thesis introduces a tractable normative framework for studying the neural implementation of computations, that we term the efficient computing hypothesis. This framework takes the efficient coding optimisation problem, and simply adds a constraint that ensure the predicted neural activity can implement the relevant computation. Most of this thesis is devoted to using this theory to normatively model two cortical representations - path-integrating representations like the entorhinal grid code, and structured working memory representations in prefrontal cortex. A minor theme in this thesis is using similar ideas to study the modularity of neural circuits. In sum, by framing a tractable normative theory for the neural implementation of computations, we take small steps towards understanding and inferring the brain's algorithms from neural activity, and predicting its puzzling neural implementations.

Impact Statement

This thesis studies tractable normative models of neural activity. It will therefore be of most relevance to two groups of people: theorists studying similar models, and experimentalists puzzling over neural circuits. For the first group this thesis contains mathematical tools that we hope will be useful, either in thinking about the neural implementation of computations, or in more standard efficient coding approaches where we provide some nonlinear yet analytic approaches. For the second group, we hope these theories will be helpful in linking puzzling observations (e.g. funky tuning curves or a cerebellar connectomic patterns) to the underlying computation. Further, we hope the underlying principles are framed clearly enough to aid reasoning about the neural circuit that interests you.

More broadly, this thesis contributes to our understanding of nervous systems, outlining links between measured neural activity and the underlying computation. One of the biggest impacts of neuroscience research is clinical. We hope that as the field progresses our improved collective understanding will pay clinical dividends, providing more tailored tools for curing neurological and psychological illnesses. Slightly more precisely, one therapeutic approach is brain-machine interfaces, which can allow paralysed people to move and mute people to speak. This involves measuring patient's single-neural activity and inferring the underlying encoding – for example, the word that the patient wants to say. This thesis contributes directly to understanding the structure of the neural code. We therefore hope that the insights gained will eventually permit our brain-machine interfaces to much more precisely decode our thoughts from neural activity.

Our work also overlaps heavily with machine learning. Most precisely, our work has contributed to two active machine learning research areas. First, we studied matrix factorisation problems, a common data analysis technique, and derived the first tight identifiability result, improving our understanding of the tool and in turn sharpening the conclusions we can draw from it. Second, ideas presented here helped in the development of a state-of-the-art disentanglement method, a nonlinear version of matrix factorisation that is used to develop interpretable AI models.

At a larger scale, the recent success of AI has led to its widespread use and influence in society. However, as discussed in this thesis, we do not understand what complex neural network models, like frontier AI systems, are doing. This lack of understanding prompts concerns, from the sci-fi 'If I want to turn it off, will it resist?' to the mundane 'why is it prescribing this drug when that would be better?' We therefore need theories to understand the behaviour of artificial neural networks. Mechanistic interpretability is an active area of work that seeks to understand artificial networks like a neuroscientist tries to understand the brain. We think a lot of our work will be directly relevant to these researchers.

Finally, and most grandly, somehow your brain creates you. Understanding 'how?' would shape our view of the world, influencing culture and ideas in the same way that our knowledge of far-off galaxies, while not directly useful, changes how we think about ourselves.

UCL Research Paper Declaration Form

Actionable Neural Representations: Grid Cells from Minimal Constraints

1. Already published research manuscript:
    - Title: Actionable Neural Representations: Grid Cells from Minimal Constraints
    - Link: https://openreview.net/forum?id=xfqDe72zh41
    - Publication name: Proceedings of the 11th International Conference on Learning Representations
    - Publisher name: International Conference on Learning Representations
    - Date of Publication: 2023
    - List of all authors: William Dorrell, Peter E. Latham, Timothy EJ Behrens, James CR Whittington
    - Peer reviewed: Yes
    - Do you retain copyright: Yes
    - Was an earlier version uploaded to a preprint server: https://arxiv.org/abs/2209.15563
2. Which chapter: chapter 3
3. e-Signatures confirming accuracy
    - Candidate signature, dated 3rd August 2025:

    - Senior Author signature, dated 3rd August 2025:

Range, not Independence, Drives Modularity in Biologically Inspired Representations

1. Already published research manuscript:
    - Title: Range, not Independence, Drives Modularity in Biologically Inspired Representations
    - Link: https://openreview.net/pdf?id=BxQkDog4ti
    - Publication name: Proceedings of the 13th International Conference on Learning Representations
    - Publisher name: International Conference on Learning Representations
    - Date of Publication: 2025
    - List of all authors: William Dorrell, Kyle Hsu, Luke Hollingsworth, Jin Hwa Lee, Jiajun Wu, Chelsea Finn, Peter E. Latham, Timothy EJ Behrens, James CR Whittington
    - Peer reviewed: Yes
    - Do you retain copyright: Yes
    - Was an earlier version uploaded to a preprint server: https://arxiv.org/abs/2410.06232
2. Which chapter: it is spread over chapter 5, chapter 6, and chapter 7
3. e-Signatures confirming accuracy
    - Candidate signature, dated 3rd August 2025:

    - Senior Author signature, dated 3rd August 2025:

Meta-Learning the Inductive Bias of Simple Neural Circuits

1. Already published research manuscript:
    - Title: Meta-Learning the Inductive Bias of Simple Neural Circuits
    - Link: https://proceedings.mlr.press/v202/dorrell23a.html
    - Publication name: Proceedings of the 40th International Conference on Machine Learning
    - Publisher name: Proceedings of Machine Learning Research
    - Date of Publication: 2023
    - List of all authors: William Dorrell, Maria Yuffa, Peter E. Latham
    - Peer reviewed: Yes
    - Do you retain copyright: Yes
    - Was an earlier version uploaded to a preprint server: https://arxiv.org/abs/2211.13544
2. Which chapter: chapter 11
3. e-Signatures confirming accuracy
    - Candidate signature, dated 3rd August 2025:

    - Senior Author signature, dated 3rd August 2025:

Acknowledgements

"No man is an island, entire of itself"
John Donne

My five years of life as a PhD student have been, by and large, a joy. I have felt free to explore, both intellectually and physically, facilitated by hordes of clever and kind people. And, when I inevitably fail, I have felt supported by a dense enough net of academic advisors, friends, and family that I have rarely felt like I slipped through its holes. For this experience I feel very lucky.

First, I am grateful to my advisors. Three advisors is a good number; though they met in the same room perhaps only once, they complemented each other very nicely. Despite endless threats of firing me, I'm grateful to Peter both for not actually firing me, and for our Friday afternoon chats which I've always enjoyed. He is endlessly willing to chat through ideas, scientific or otherwise, and from his unwavering questioning I think I've become clearer in my talks and writing. Tim welcomed me into his lab's exciting mission where I have enjoyed feeling part of something bigger than my little project. I've learnt a lot from his deep intuitions for both neuroscience and linear regression, and I've enjoyed the infectious zappy excitement of working through ideas together. James provided me with my first PhD project, the grid cell work presented here, and has been involved with almost every step since. He has answered my endless inane WhatsApp messages, ranging from technical to political, and his guidance has made me a much better scientist. I thank all three of them for devoting so much time to helping me.

More broadly, I've been learnt a lot from the Gatsby Unit which has been an intellectually lively and impressive place to work. I'm grateful to all the members of Gatsby, Tim's Lab, and the SWC for providing such a stimulating place to do a PhD. Of particular note are the brave (foolish?) people who collaborated with me. I had fun hacking around with the PPSeq team – Tom George, Rodrigo Carrasco-Davis, Clementine Domine, Emmett Thompson & Marcus Stephenson-Jones – finding evidence of replay with the team in Emmett's pristine data at the end of my first year was a zinging feeling that was all the better for sharing it with a great team. Maria Yuffa made the bold decision to work with me during second year while still an undergraduate and I'm grateful for all the trust and dedication she put into our project. Finally, I had a lot of fun cos-playing as an experimentalist during the last year of my PhD, and for that I am very grateful to the amazing grid cell team – Ryan Cini, Francesco Pozzolo, Peter Doohan, Charles Burns, and Beatriz Godhinho (and Vasco for helping me saw polystyrene balls with a bread knife – his lungs' loss was sciences' gain). Their dedication and skill were awe-inspiring, and they stepped up even while my brain was out of order with covid fog.

One of the big perks of being a PhD student has been the ability to travel. Conferences and research stays have made the last 5 years an extremely rich experience. I'm very grateful both for the surprising number of places I've been able to chat science, tap on my laptop, or scratch in a notepad, and for the chance to remain not doing science afterwards. I spent a happy 3 months at Stanford working with Kyle Hsu, the ideal collaborator – patient, smart, and frank when I'm being an idiot – and I'm grateful to Kyle, the Finn lab, Rylan Schaeffer and especially Apoorva Rangan for making that period so enjoyable and showing me some of California. I had a fun two months in South Africa, teaching on the Imbizo, hiking Table Mountain 14 ways, and working with Devon Jarvis at Wits – Devon is impressively hospitable and his friends, Geraud Nangue Tasse and Branden Ingram, were very friendly indeed.

But, the biggest reason I have enjoyed the last five years has been my wonderful friends and family. The PhD introduced me to people who are now some of my closest friends. Pierre is the perfect person to chat with long into the night and I value deeply his wit - which keeps me honest – and his company - which keeps me happy. Rodrigo is like a chuckling Buddhist sage; a rock in any storm with wise counsel, and a willing partner in exploring the world's weirdnesses. Clem's bright and friendly light makes any situation fun, and I've enjoyed gossiping in the office or adventure to chatting with her grandma or drinking spritzes around Rome. Further, I'm grateful to the many friends I've hung out with during this time, Sebastian, Sarah, Tom, Peter V, Basile, Jin, and many others. I feel very lucky to have you in my life.

I'm grateful to friends from other parts of my life - meadhurst, undergrad, Boston, & Okinawa - for helping me enjoy to life and for making it interesting. Sadly, their contribution is out of scope of this work, but it would be remiss not to mention some highlights, like Rob's musical extravaganzas, the crazy cats who helped me walk Offa's Dyke, Charlotte's writing retreat, the most amazing 2 month road-trip with Lorena (& goma!) that showed me how big the world can be, and the kindness of Chapi & Vega (& hector & Luna) for hosting me in style multiple times. I should, however, thanks my flatmates of four years – Lili, Eliot, and Tom – for putting up with me; special mention goes to Tom who has managed 10 years of overlapping both his work and home life with me – and whose company throughout these years I value greatly.

Finally, I would like to thank my family whose love and support I cherish. I would not be here without them.

It warms my cockles that there are so many wonderful people in the world and that I get to spend time with them.

Contents

Chapter 1

Introduction

"Nothing makes sense in biology except in the light of evolution"
Theodosius Dobzhansky, 1964

It requires an uncommonly pedestrian outlook to avoid being awed by Nature's ingenuity. This thesis studies nervous systems, the evolutionarily ancient control systems shared by almost all animals (except those pesky sponges). The function of nervous systems is to control mobile multicellular organisms. This function is well illustrated by sea squirts (*ascidians*). Juvenile sea squirts are mobile tadpole-like creatures complete with sensors for gravity and light, muscles, and a nervous system to coordinate them. But in their adult form sea squirts settle down to a sessile life of filter-feeding; this lifestyle doesn't need a nervous system, so sea squirts recycle theirs in the transition to adulthood.

Nervous systems are the source of many of Nature's most awe-inspiring displays. They permit flies to fly, gymnasts to gyrate, ant colonies to farm aphids, or orca pods to learn culturally specific hunting practices. They construct sensors for every conceivable input – vast swathes of the electromagnetic or vibrational spectrum: from low rumbles in elephant's feet to 300kHz in the ears of bat-avoiding moths. They analyse these signals – distinguishing predators from prey or lovers from lunch, coordinate exquisite movements, remember across lifetimes, construct cultural histories across millennia, and form our internal worlds and sense of ourselves. And they do all this using a self-organising tissue that uses less energy per gram than your heart (Sterling and Laughlin, 2015).

Our question, of course, is how?

Answering this question is an enormous ongoing endeavour. On the top end, cognitive scientists, psychologists and ethologists try to describe the algorithms at work in our nervous systems by probing behaviour. Perhaps your dog, after a painful experience, learns not to run into a glass door; then you can play the dognitive scientist and infer the existence of a learning algorithm that must be implemented somewhere in their brain. At the other end, all manner of biologists are at work describing the biochemical underpinnings of these algorithms, examining the chemicals and proteins that compose our cells and endow them with their abilities.

Progress on all levels will be necessary to understand nervous systems. In this thesis we will be situated somewhere in the middle of these two extremes, with a slight bias upwards, an area called systems neuroscience. Our base will be neurons and how they behave. Neurons are the fundamental unit of the nervous system – a cell specialised for communicating signals over long-distances using electrical currents. Many impressive organisms don't have neurons, for example there are numerous clever single-celled organisms that show adaptive behaviours using chemical signalling. But as soon as your body becomes large, chemical diffusion is slow, and coordinating your behaviours requires quick, precise signalling. Neurons provide this, allowing an impulse in your brain to reach your toe in 10ms, pleasingly curling it (try this now – wasn't that impressively quick?).

The language neurons speak is, approximately, electrical, and that language is shared across the nervous system. Neurons discharge spikes, rapid electrical transients, up to 100s of times a second, which are then funnelled to their appropriate targets via a cable - the axon. Neurons in the eye use these spikes to encode patterns of visual inputs, those near your muscles use them to control movement, while frontal cortical spiking stores your understanding of the transient state of the world. In all cases, neurons encode meaning using patterns of electrical activation. Just like electrical activity in a computer, what differs is the patterning and its assigned meaning, not the substrate.

This thesis will focus on the link between patterns of neural activity and the algorithms implemented by

neurons. Inevitably, ignoring lower levels of description (ion channels, vesicles, glia) will introduce errors. We'll even simplify the neurons' electrical code. For example, we'll ignore what is perhaps neural activity's most apparent feature, it is a punctate sequence of spikes. Instead, we'll imagine what matters is how much each neuron spikes in a small window of time, a firing rate code (it is an ongoing debate how reasonable this is (Gautrais and Thorpe, 1998; Schmutz, Brea, and Gerstner, 2025)). These assumptions are a shame, but that's just how it goes. Understanding involves bridging levels of description, and patterns of neural firing rates and their algorithmic function seem a reasonable choice.

This thesis is about normative models of neural activity, and neural circuits more broadly. We'll next describe what that means, and how this approach is situated relative to the rest of neuroscience, before zooming into different normative approaches, and the approach this thesis will mostly take.

1.1 How might we go about understanding a nervous system?

A first attempt to understand nervous systems might involve measuring all the neurons at once and examining how it allows the animal to behave. This is not just a thought experiment: this can be done with small organisms, including perhaps soon the fruit fly. However, this is difficult. Both the algorithm and its neural activity are complex: without a lot of work, the algorithm is just a loose injunction to 'behave!' and you want to tell me how it emerges from the fruit fly's millions of co-activating neurons. Sounds difficult in one jump.

Instead, to make the problem tractable we exploit the pervasive modularity of nervous systems. Brains contain highly specialised regions: a small bundle of neurons deep in your brainstem regulates your circadian rhythm, while small parts of cortex let you read, or recognise faces (the Fusiform Face Area – FFA). Normatively, this makes sense: your brain is under intense evolutionary pressure to be small (births are already difficult enough). Connecting neurons together requires a lot of internal wiring, which uses a lot of space and energy. Therefore, neurons are structured to minimise wiring costs: they communicate mainly with their physical neighbours using short wires. This means that meaningful sub-computations are likely performed in small regions, allowing us to measure from a small area and hope to find neurons performing a meaningfully understandable computation, which can then be integrated into a larger picture. This gives us hope that the full complex problem breaks down nicely, allowing us to work in parallel on what different circuits or brain areas are doing.

So, you then want to understand how neural activity in one brain area subserves its paired function; how might you proceed? Getting a full picture requires integrating diverse streams of information, we'll give a few examples here. There is, however, no clear flow. Each stream of information informs the others, for example, linking function to anatomy will make it clearer what are the meaningful divisions of cortex. As such, these processes will all operate in tandem, hopefully iteratively converging on a better understanding.

Anatomy can tell you some things. The retina is structured in three clear layers of neurons mapping from initial receptor measurements to a large output bundle – the optic nerve. It is clear there is some iterative computation occurring on visual information before transmission of the information to the rest of the brain. Similarly, different cortical areas show clear anatomical differences in cell counts, structures, and layering, that must, in a currently somewhat mysteriously manner, be linked to their function.

Further, various techniques can be used to pin down the algorithm of a brain area more precisely. One seam of evidence comes from patients with localised stroke damage leading to surprisingly specific behavioural deficits. A famous example is 'the man who mistook his wife for a hat' because a stroke in his FFA meant he no longer recognised faces, clearly linking the damaged brain area to facial recognition (Sacks, 2011). Or, rather than waiting for a precise stroke to occur, the invention of fMRI 35 years ago allows you to coarsely measure the brain activity inside behaving people (as long as the behaviour takes place inside the small noisy tube of the machine). You could design a battery of tasks with different algorithmic requirements - perhaps with or without facial recognition - and infer from the pattern of brain activity which area was involved in facial recognition (Kanwisher and Yovel, 2006).

Finally, comparative studies can describe analogous brain regions or behavioural patterns across animals. You can then relate the structure of the region to the variable behavioural demands placed on different animals. For example, two fish species of similar sizes have brains that differ in size enormously, largely due to the presence of an expanded cerebellum, fig. 1.1. This can be explained ethologically. Mormyrids live in turbid water with low visibility, so they invest in a large electrosensing system for catching prey, whereas trout live in clear water, and don't make this investment. In the species that have been studied, analysing electric field data seems to require a large cerebellar-like structure.

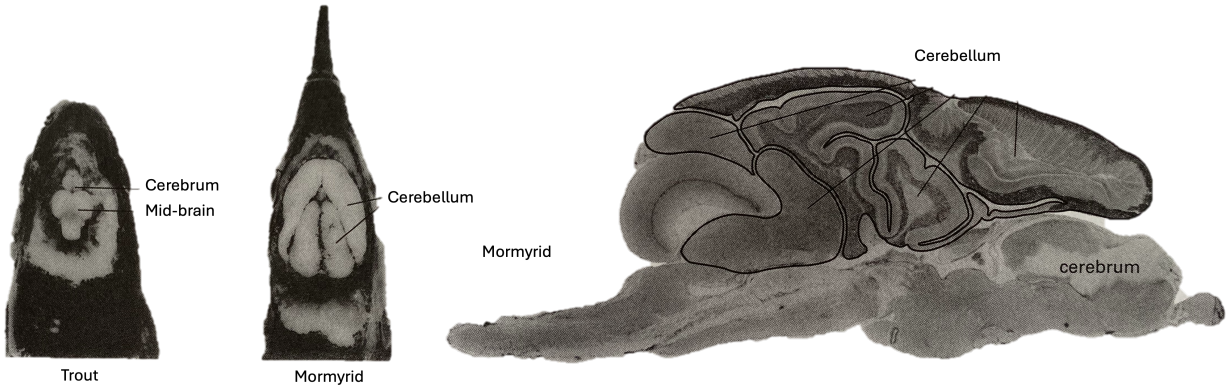


Figure 1.1: On the left are the brains of two similarly sized fish, note the enormously expanded cerebellum in the mormyrid to process electrosensing data. On the right is a cross-section of the mormyrid brain, showing that it is completely dominated by cerebellum (Nieuwenbuys and Nicholson, 1969)

But to understand how neurons produce behaviour we eventually need to measure the activity of single neurons in the relevant area. To relate these neural measurements to function requires a model that explains how the observed neural response subserve the proposed algorithm.

Models come in a variety of forms. Two broad categories are mechanistic and normative. Mechanistic models are bottom-up. They attempt to simulate a circuit of plausible neurons that behaves similarly to the measurements you make while performing the function you think is relevant. These can be compelling. For example, one puzzle is presented by early visual processing in cortex, which is thought to extract meaningful components of images, like the presence of lines. In particular, edge detector neurons, that activate when there's a line at a particular orientation and position in the image, were discovered in visual cortical area V1 of cats in the 1960s (Hubel and Wiesel, 1962). As an early example of non-trivial cortical processing in awake animals this was a foundational moment for neuroscience. In thinking about their function, early mechanistic models were sketched out. Retinal ganglion cells, the neurons that make up the optic nerve transmitting information from the retina to V1, were known to signal the contrast of a small region of the visual field; i.e. each ganglion cell might encode the presence of a bright patch at a particular position. Early models demonstrated how a neuron in the visual cortex could construct an edge detector from this by summing the activity of a sequence of such retinal ganglion cells arranged in a line. The existence of such a plausible model lends credence to the entire theory of edge detection in early visual cortices.

However, this thesis is about the other type of model: normative or top-down models. Rather than constructing a set of neurons that do something in a plausible manner, we begin with the proposed function, which can be surprisingly general. We then reason about how performing this function, subject to various constraints, leads us to the observed neural activity. Sometimes the distinction between normative and mechanistic is not so clear. One pertinent contrast between extremes might be that in mechanistic modelling you often try to construct a model that fits the data, leaving open the possibility that there might be other reasonable models you haven't considered. Normative modelling, by framing the problem as an optimisation, forces you to consider all the models that fit your framework. A common experience is imagining one circuit to be very good at a particular task, but then being told by your normative model that another was much better, highlighting parts of your problem that you didn't consider.

Since they form the topic of this thesis, we'll spend a while describing them in more detail. But to close this summary of the broad approach of systems neuroscience, by integrating normative models with findings from across the streams of evidence listed above, concrete progress can and has been made in outlining the link between neurons and behaviour, as in the V1 example above.

1.2 Normative Models of Neural Activity

As the quote at the beginning of this chapter outlines: the structuring principle of biology is evolution. Biological structures exist to permit survival and reproduction. As such, a trivial answer to why is the brain like this and not like that is because it helps us survive. But, framed at this level of generality, this is not particularly useful. Surviving is such a nebulous concept: trees survive by growing tall and sprouting solar panels, whales by filtering the ocean for krill; both are clearly successful. With such a diversity of approaches available it becomes hard to clearly link a puzzling piece of neural structure to survival.

Instead, we imagine useful proxies. We discussed one earlier, minimising wiring costs. Normative arguments based on wire go as follows: all else being equal, it's better to use less wire, since its construction and operation

is energetically costly, and space is usually at a premium. Using less wire ensures an animal can survive on a little less food, making it more likely to succeed. Further, this proxy is framed at a useful level. We mentioned the Fusiform Face Area which performs facial recognition. Recognising a face requires access to both high level visual processing and recall of factual memories associated to the face in front of you ('That's my boss – look busy!'). This can explain why, to minimise wire, the FFA is situated close to the top end of the ventral visual stream, which performs detailed object identification, and near the hippocampus, which permits the recall of memories.

Minimising wire is so broadly useful in understanding the structure of the nervous system that it warrants denoting as a principle. As in the case of the FFA, at the macroscale it explains the broad placement of different brain areas based on their function (Cherniak et al., 2004), or the separation of the brain into white and grey matter using conduction delay ideas (Q. Wen and Chklovskii, 2005). On the mesoscale it explains why some areas of visual cortex show blob like spatial patterns of tuning to the two eyes, while other parts are striped (Koulakov and Chklovskii, 2001). And on the microscale, it explains the placement of neurons into layers (Sterling and Laughlin, 2015), or the widescale use of dendritic spines to facilitate dense inter-connectivity (Chklovskii, Schikorski, and Stevens, 2002).

These arguments are deeply satisfying. However, they can whiff of just-so stories. It is relatively easy to come up with plausible sounding explanations for various evolutionary choices. How can we arbitrate between explanations that are useful rather than just satisfying? For example, the common, and satisfying, explanation is that giraffes evolved tall necks to browse higher on trees, but a more successful theory would explain why such animals did not evolve elsewhere in the world despite the pervasive existence of tall trees with nutritious leaves. (In fact, it seems likely that giraffes evolved long necks not primarily to browse, but due to sexual pressure, explaining why other animals did not follow the same path (Simmons and Altwegg, 2010; S.-Q. Wang et al., 2022)). We argue that two aspects of the wire minimisation principle make it an emblematic success story for normative theorising: it is framed mathematically and generally.

First, it is relatively easy to frame a mathematical version of the wire minimisation argument. For example, say you have a set of neurons and their connectome – a list of which neurons are connected. You can imagine neurons as points in 3D space, and their wiring cost as the sum of the distances between all connected neurons. Wire minimisation would predict that the brain should arrange the neurons to minimise this cost. This is a clean prediction that, in *C. elegans*, a small worm, fits relatively well (B. L. Chen, Hall, and Chklovskii, 2006). In general, a mathematical framing makes predicting simple - frame the optimisation problem and study the solution. Further, by framing the problem mathematically, you're forced to concretise potentially loose ideas, often in ways that teach you about the structure you're studying[1]. Life is never this simple, for example it is not obvious that a wire that is twice as long is twice as costly, in fact there's evidence that costs scale as the square of the length, rather than linearly (Perge et al., 2012). But in broad strokes, the problem and its prediction are often clear.

Second, the wire minimisation principle is incredibly general. It can be applied to all nervous systems. This means we can develop the theory in one setting, tuning things like whether cost is linear or square in length to fit one set of observations, then predict in a new setting. So not only does the theory's broad applicability make it more useful, but it also means we can test it. Finally, not only does wire minimisation regularly explain new findings, it can be used in reverse. If you find a new brain area and don't know its function, you can look at the neighbouring brain areas and, because wire minimisation is so well substantiated, infer that the function will likely have something to do with information encoded nearby.

As such, normative approaches can provide structuring principles that explain and predict observations across nervous systems, and open new lines of evidence for understanding the function of less understood circuits. In this thesis we're mostly concerned with neural firing and its relationship with computation. We will therefore turn now to normative theories of neural activity. We will discuss the paradigmatic example, the efficient coding hypothesis, and a long-standing alternative, connectionism, before explaining where the work in this thesis sits relative to these.

1.3 The Efficient Coding Hypothesis

In the 1950s and '60s neuroscience was performing the first recordings of cortical neural activity in behaving animals. This lead to foundational discoveries, like edge detectors, neurons in V1 that activate to the presence of an oriented edge. Concurrently, people began asking the natural normative questions: these neurons are clearly encoding the visual scene, but why in this way? For example, why edge detectors and not block detectors?

This early work was quick to use the then recent developments in communication theory to answer these questions. The development of computers in the '30s and '40s had prompted a mathematically study of the quality of different codes (Shannon, 1948). For example, perhaps you want to send a text message across an

[1]That said, this is not an argument for mathematically pinning down everything. Some ideas clearly have power, but lose it when mathematised. Just because an idea is difficult to formalise doesn't make it useless, it is just better if it can be.

undersea cable. To transmit it across the cable you need to turn each letter into a binary pattern (e.g. a = 0000, b = 011, c = 10101 ...). Any encoding that assigns a different code to each letter will work, so you need at least 26 codes. But some of these encodings, like using hundreds of bits for each letter, are clearly suboptimal. Space on the cable is at a premium, so you want the code which uses, on average, the smallest number of bits to send each text. Framing and solving this problem led to a mathematical formalisation of the idea of information. Roughly, if a letter occurs often (like 'e' in English) you should give it a short code (perhaps 00), whereas a rare letter (like 'z') should get a longer code (010110). Information theory tells you exactly which types of codes you should use to minimise the cost of sending a message (or, equivalently, maximise the amount of information carried by the code in a given time interval).

The analogies to neuroscience are clear. The retina is producing encodings of images and sending them via a costly undersea cable (the optic nerve) to your visual cortex. Therefore, retinal ganglion cells should structure their encoding of information to maximise the amount of information it can transmit per unit time, subject to the constraints of an optic nerve bottleneck. This logic, first framed in the 50s and 60s (Attneave, 1954; Barlow et al., 1961), is called the efficient coding hypothesis: the idea that neural firing can be understood as an optimal encoding of information for communication. This theory is beautiful and predictive; and as we'll now explain, it matches both our previous requirements for a normative theory.

On the one hand, mathematically formalising it is relatively easy. Choose a set of variables to be encoded (e.g. letters in a text, or images on a retina), create an encoding and measure its quality, for example using the mathematical formalisation of information cited above. Finally, introduce the constraints on the neural code – for example firing rates are non-negative (since you can't have a negative number of spikes per second), and you might constrain the code to have a limited total amount of firing, to save energy. Then optimise the quality of the neural encoding of images subject to these constraints.

On the other hand, it is general enough that it can be used to model neural firing basically anywhere, and To model a different circuit you must simply change the variables being encoded. Further, . One clean prediction comes from efficient coding's dependence on the statistics of the variables encoded. For example, if you switch languages 'e' might suddenly occur rarely and 'z' often, so you should change your encoding to maintain efficiency. This is very testable: animals that experience different sensory distributions should use different encodings and these patterns should be understandable from the efficient coding hypothesis.

The efficient coding hypothesis has been highly successful. Versions of it can explain why auditory neurons fire at higher rates than visual, which fire at higher rates than olfactory sensory neurons (Perge et al., 2012; Sterling and Laughlin, 2015); the distribution of cone types on the retina (Gupta et al., 2023); why bipolar cells have OFF and ON types (Gjorgjieva, Sompolinsky, and Meister, 2014); or why retinal ganglion cells have centre-surround receptive fields (Atick and Redlich, 1990, 1992), among many other things. Let's zoom into one example. A particular version of the hypothesis – sparse coding – can answer the question posed at the beginning of this section: why is V1 full of edge and not block detectors? In sparse coding models, you optimise the neural encoding of natural images (like a forest, rock, or stream) in order that it is a maximally faithful - i.e. you can reconstruct the image from the neural activity relatively easily, while minimising the amount of neural firing. Doing this leads to neurons tuned to edges, just like in V1 (Olshausen and Field, 1996) (and in fact, this result can also be shown from a pure information maximisation perspective (Okajima, 1998a,b)). As such, this is another normative principle that has proven its worth time and again. Indeed, in part II of this thesis we will develop efficient coding style arguments to understand the emergence of functional cell types and mixed selectivity.

Yet, broadly, this thesis argues against using the efficient coding hypothesis as a model for most neural activity, especially in cortex. In the next section we will outline this argument using the example of grid cells.

1.4 Cortical Activity is not *just* an Efficient Code

Despite the elegance of the efficient coding hypothesis, let's pause to evaluate it for a moment. The theory is based on the premise that neural activity is forming an optimal encoding of information for efficient transmission elsewhere. There are clearly places where this mental model seems perfect, for example sense organs are often physically displaced from the cortical areas that analyse their output, requiring cables, like the optic nerve, to transmit the information. In these cases, viewing neural firing as transmitting an encoding from sense organ to cortex through a limited bottleneck seems well justified. This view is further bolstered by the fact that neural firing in these areas can often be well-understood using efficient coding arguments, as outlined above.

However, most of the brain is not in the game of summarising and transmitting information. Rather, it must compute. The variables that arrive from our sensory inputs are not ready for driving behaviour, hence there are large chunks of cortex devoted to further processing each sensory input. For example, visual cortical areas are, in ways that are not well understood, sticking together the pattern of input edges and the broader context to understand the objects in the world around us (round, no spout or handle, and I'm on a football field = football; round, a spout and handle, I'm at my grandma's = teapot). They will then transmit their finding

(football vs. teapot) to other relevant cortical areas, likely using an efficient code. But in visual cortical areas we should expect to find neural activity that reflects the ongoing computation, the inference of the object.

Let's make this argument more precise using the most well-understood circuit in mammalian neocortex: the grid cell system (the topic of part I). Grid cells are delightful neurons that encode an animal's position in space, fig. 1.2. When an animal runs around a flat 2D room, each grid cell activates in a hexagonal lattice of positions (Hafting et al., 2005). Further, and crucially, grid cells come in groups called modules (H. Stensola et al., 2012). All grid cells in the same module have receptive fields that are translated but not rotated versions of one another – they share the same firing lattice. There are a handful of modules in each animal, and each module contains hundreds of neurons, which together tile all possible translations of the lattice, fig. 1.2.

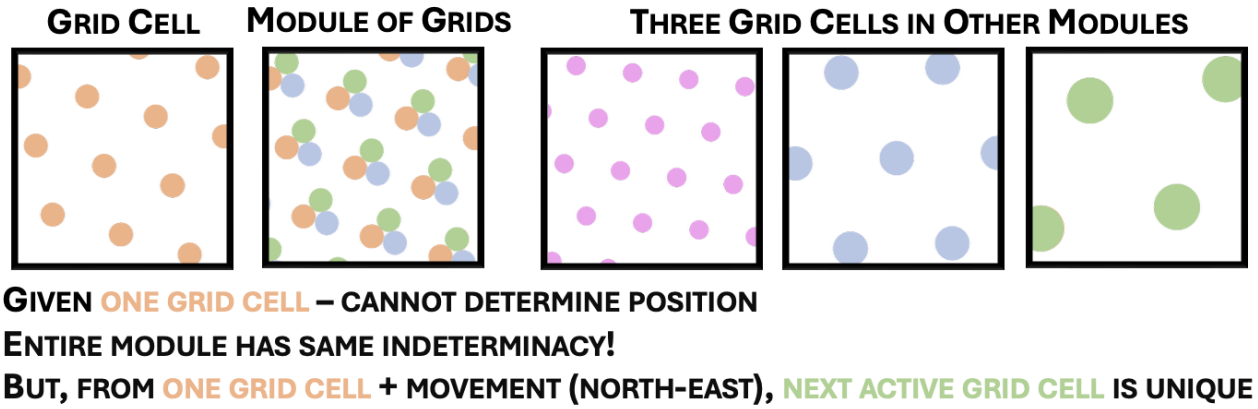


Figure 1.2: Left: one grid cell activates when the animal is in a hexagonal lattice of positions in the room. Middle: there are many grid cells grouped into modules. Grid cells in the same module have firing patterns that are translated versions of the same lattice. The blue, green, and orange firing patterns show the firing patterns of three different grid cells in the same module. Right: three other grid cells in different modules. The text highlights that if you only know the orange neuron is active you could be in any of its firing fields. The other neurons in the same module will not help you work out where you are (though the other modules will). However, the other neurons in the module make one thing very easy: if you only know the orange neuron is active, and then you step north-east, you can predict that the green neuron in the same module should activate. The grid cell system is not optimal for efficient coding, but it is optimal for path-integration, and the neural activity tells us that.

This is an incredibly structured set of neural activities, prompting obvious questions: what is it for? Since it is clearly tuned to the animal's location, a natural normative approach is to attempt to understand this representation as an efficient code for position. Indeed, the grid cell code is quite a good encoding of position. But it has one clear design flaw. Each grid cell is roughly symmetric, meaning it encodes positions separated by a lattice symmetry identically – if one grid cell fires you can't tell which firing field you are in. If the neural encoding was truly an efficient code for position, other neurons would structure their firing to remove this degeneracy; perhaps a few other neurons would each fire in one of the grid cell's fields, so that from the combination of which two neurons were firing you would know where you are. Yet, rather than doing that, the nervous system 'wastes' an enormous amount of energy creating a whole module of 100s of neurons that share exactly the same lattice symmetry, the same degeneracy in encoding. This is not an efficient code! Indeed, as we explain in chapter 4, despite 15 years of concerted effort, no efficient coding theory for grid cells is able to explain the modular periodic structure observed without baking it in.

Instead, as we also review in chapter 4, both mechanistic work and various lesion studies present strong evidence that this circuit performs a computation called path-integration: it integrates your sequence of movements to keep track of where you are in space. For this computation, the aligned modular encoding is perfect. If one grid cell fires I may not know exactly where I am, but I will know that if I step right the next grid cell that should fire is the one with a tuning curve translated one step to the right, fig. 1.2. Hence, this puzzling representational property is only puzzling from an efficient coding perspective, from a navigational perspective is has long been clear that this representation is excellent (McNaughton et al., 2006; Samsonovich and McNaughton, 1997).

This illustrates the broader point (perhaps the one paragraph take-away of this thesis): cortical neural activity is subserving computations, like path-integration or inference of object identity. When we measure neural activity, we are measuring the fumes of ongoing computation, not the passage of carefully formatted messages between unspecified observers. We should expect to see signatures of these ongoing computations that do not make sense from an efficient coding perspective. Therefore, we should be designing normative theories

that link the computations happening in cortex to their neural underpinnings. This is the broad topic of this thesis. Before describing these approaches in detail, we contrast it with the other major normative approach.

1.5 Connectionism: Modelling Neural Activity with Task-Optimised Artificial Neural Networks

Artificial neural networks have become rather popular lately. By training the parameters of large networks to perform tasks on enormous datasets (i.e. the internet) these networks can learn to chat like people with immediate access to the entirety of Wikipedia, or to recognise objects in images – something previously only the visual cortex was robustly able to do.

This development has been especially interesting for neuroscience. Even if the network is treated as a black-box that swallows images and spits out object identities, it is already transformational to have a model that can do these complex tasks. No previous model comes close to the performance of either artificial neural networks or the visual cortex. This allows us to probe them behaviourally. For example, you might be interested in how humans learn to recognise objects, now you can test hypotheses by training networks on different datasets and seeing how this determines the model's ability to identify novel objects. The assumption, likely partly correct, is that there are general principles that apply to any learning system performing the same object detection tasks, including both artificial and biological neural networks. Since testing an artificial network is often much easier and can be done much more thoroughly and precisely, we can hope to learn lessons from the artificial neural networks that generalise to the brain.

However, these networks are additionally interesting, not just as a black-box performer, but as a model of neural activity. Artificial neural networks are modelled on the nervous system. Their fundamental unit is an artificial analogue of the neuron that integrates the outputs of other neurons and contributes its own output to the mix. Just like the brain, artificial networks trained on object recognition perform iterated layers of neural processing turning input images into increasingly abstract representations and eventually, an object categorisation. Indeed, neurons early in the process behave just like the neurons we've discussed in early visual processing in the brain, including centre-surround and edge detecting neurons (G. Hinton et al., 2006).

Therefore, artificial networks present an alternative approach to building a normative model of neural activity, known as connectionism. Step 1: frame the computation you think the brain is performing, step 2: train a neural network to do it, step 3: compare the neural activity to that measured in the brain. The evidence suggests that this has had broad success across visual (G. Hinton et al., 2006; Yamins et al., 2014), auditory (Kell et al., 2018), prefrontal (Botvinick and Plaut, 2006; J. X. Wang et al., 2018), motor (Tanaka, 2016; Zipser and Andersen, 1988) and language (Gauthier and Levy, 2019; Goldstein et al., 2022) cortices (and likely many others I'm not aware of). While connectionism has existed for at least half a century, the huge recent developments in machine learning have made it much more prevelant. Indeed, training an artificial neural network model and comparing it to neural recordings is now a standard part of experimental neuroscience papers. As such, this approach seems perfect – it examines the behaviour of neurons performing meaningful computations and in so doing can regularly match measured recordings, avoiding the pitfalls of the efficient coding hypothesis and its progeny.

Unfortunately, however, connectionism suffers from one major downside. Artificial neural networks are often inscrutable; on their own, they offer little understanding about why neurons are behaving as they are. This is worrying on multiple levels. First, human understandable pictures are a large part of the output of the scientific process. Barring sci-fi futures involving wholesale replacement of science by AI, our goal is to construct reasoning that is sufficiently predictive to be useful and sufficiently simple to be shared; a basis on which we can construct a robust understanding of the world around us. If such a communicable, predictive picture can be constructed it seems preferable to a black-box predictor. It may well turn out that there is no simpler solution to understanding neural activity than training a neural network, but I would suggest the evidence doesn't support this view, warranting further investigation of simpler models.

Secondly, and more concretely, it can be easy to build models that somewhat work for unknown reasons, tricking ourselves into simple stories. Grid cells again provide a good example of this. Two concurrent papers trained networks on tasks that involved path-integration and looked at the networks' internal representations, finding grid cells (Banino et al., 2018; Cueva and Wei, 2018). This is potentially a slam-dunk, leaving us with a simple interpretable story: networks trained to path-integrate need grid cells. However, it turned out to be more complicated. One approach used a large network trained on a complicated reinforcement learning task and reported hexagonal grid cells (Banino et al., 2018), yet it was later discovered that the 'grid cells' reported were as griddy as low-pass filtered noise (Sorscher, G. Mel, et al., 2019), and they lack the axis-aligned modular structure observed. They were only superficially grid cells. The other paper trained a much simpler network and observed modules of square rather than hexagonal grid cells (Cueva and Wei, 2018). This result was robust (motivating the use of simple networks and tasks) but why the grids were square and not hexagonal is not obvious from the modelling work, making it hard to know what to change to develop a better model. We will

review this literature in full in chapter 4, but suffice to say, deep networks did not prove a panacea. A robust task-optimised model of grid cell was only developed after significant additional theoretical work with simple models.

As such, this lack of explainability is deeply dissatisfying. We've simply replaced one set of confusing neurons in the brain with another in an artificial neural network, leading potentially to errors, and making it hard to understand what to change in order to make our models better. That said, I fully believe that task-optimised neural networks are an excellent addition to the toolbox of computational neuroscience. Robustly demonstrating an artificial analogue of a set of biological neurons is excellent news: now you can probe them in silico without any of the time or mess of real biological experiments. For example, you can test whether edge detectors are really a property of natural images by training the network on images with very different properties – like a preponderance of curves – and see if it learns curve detectors instead. If so, this presents evidence that edge detectors really reflect a sparse code of natural images which would be hard to acquire biologically.

So, while it is exciting to be able to capture neural behaviour in task-optimised neural networks, their lack of explainability make them prone to scientific errors and difficult to understand. These are not insurmountable obstacles, but they turn understanding artificial neural networks into its own important scientific problem; one that is potentially a stepping stone to understanding the brain circuits they model. This difficult problem has to be attacked with the same suite of tools we described in neuroscience: careful experimentation to check the robustness of the results, mechanistic modelling to explain subcircuits, anatomical tracing, perturbation studies, and, of course, (normative) theorising to explain why some networks and not others learn the representations we measure. It is within this last category that this thesis aims to make small steps.

1.6 The Efficient Computing Hypothesis

To summarise our argument so far, we would like a tractable normative theory for neural activity engaged in meaningful computations. The efficient coding hypothesis is beautiful and often predictive, but its emphasis on communication leaves it badly suited to studying the properties of networks of neurons performing computations. The alternative, connectionism, can study neurons performing almost any computation we might imagine; but its inexplicability leaves us in a structurally similar place to where we began - needing theories to explain the measured patterns of neural activity. Therefore, this thesis attempts to build tractable theories for neurons that compute in some simple but useful settings. We term this approach the efficient computing hypothesis - the simple idea that neural activity can be understood as an efficient implementation of the required algorithm. We will now explain our technical approach in broad strokes.

We begin with a similar optimisation problem to those studied by efficient coding approaches: we consider a population of neurons encoding some variables, and construct an optimisation problem that has two parts. One, a functional part, chooses a high-quality encoding of the variables, just like the informative encoding part of the original efficient coding hypothesis. The second, a biological part, is similar to the efficiency ideas in a standard efficient code; it makes sure the predicted firing rates are reasonable (i.e. nonnegative but small). Various choices of these functional and biological terms create different efficient encoding problems that have long been studied in the literature.

To this standard setup we add an additional set of constraints that we've been calling 'actionable'. These enforce the computing properties of the network, ensuring that the network actually implements the algorithm in question. For example, if you are studying a path-integrating circuit, these will ensure that the predicted neural representation can be used as a circuit for tracking your position. By making reasonable choices for the functional, biological, and actionable parts of this optimisation problem, we create optimisation problems that are both tractable, and able to capture the interesting structure in measured neural representations.

Our model bears some similarities to theoretical work in the machine learning literature. Indeed our optimisation problem can be understood as studying a particular neural network architecture - a gated recurrent linear neural network. Given our motivation as developing tractable versions of connectionism, our proximity to machine learning is by design. That said, our winding neuroscientific motivation means we diverge from the approach of most machine learning theory. In particular, we do not consider learning, the target of most theoretical investigation in machine learning theory. Further, our focus has been highly neuroscientific, making some of our explorations slightly obtuse for a machine learner. As such, this thesis is a neuroscience thesis, and we delay drawing more extensive parallels to machine learning theory until the conclusion.

In sum, by combining a computability constraint with standard efficient coding approaches, we are able to construct tractable theories of neurons performing meaningful computations. In particular, we use these approaches to understand the grid cell representation of self-position, part I, and the representation of structured working memory in the prefrontal cortex, part III. These steps are small, but we hope that the evidence of their efficacy will become clear, and that future work will enlarge these normative bridgeheads into other cortical computations. We conclude by outlining the structure of the thesis.

1.7 Outline of Thesis

In part I we study path-integration. We begin by motivating and framing more precisely what we mean by the efficient computing hypothesis in chapter 2. We then use this theory to build a normative model of the grid cell representation, chapter 3. Finally, we conclude by summarising all normative work on grid cells and what we can learn from this research area with a happy combination of mature theory and precise electrophysiology, chapter 4.

Then, in part II, we take a tangent. As expressed earlier, the approach we are advocating only works if the brain is modular. This is what let us decompose the problem into manageable parts. However, recent findings highlight that the brain is not always so understandably modular, especially on the neuron level. We therefore turn to a simple efficient coding model to understand when we should expect representations to be modular, chapter 5, at times retro-justifying our previous approaches. The move to an efficient coding approach is a shame, we'd rather an efficient computing one, and in chapter 6 we make some limited progress at generalising these results to a computing setting, i.e. a recurrent linear network rather than a feedforward linear network. Next, in chapter 7, we use these results to understand some puzzling patterns of modularisation in grid cells. Finally, in chapter 8, we build a large family of convex efficient coding optimisation problems, and use these to derive more general modularising results, including a normative explanation for the presence of OFF and ON channels in retina and elsewhere. We think this last part will be of broad technical use to people working in normative representational theories and matrix factorisation problems.

Having drunk deep on more technical problems, in part III we return to the second edition of the thesis' core argument. We turn to structured working memory representations in prefrontal cortex and develop a similar efficient computing theory that can explain the patterns of measured neural activity.

Finally, in our last section, part IV, we make an abrupt shift. We turn to systems where the computation involves learning an association between inputs and outputs, in particular, cerebellar-like circuits. We study how structures in these circuits - particular connectomic patterns - are reflected in the associations that can be learnt using a small amount of data. In chapter 12 we use recent machine learning theory studying kernel regression to describe this relationship between structure and function and use it to explain these connectomic patterns. Relatedly, in chapter 11, we develop a flexible meta-learning approach that can build the same link without the need for the analytic tractability of kernel regression.

Part I

Path-Integrable Representations

Chapter 2

Normative Theories of Group-Structured World Models

We argued that normative theories in neuroscience have often been surprisingly passive. They have focused on efficient encodings, an information theoretic view which places paramount the effective communication between parties. This view produces beautiful theories that seem well suited to cases like the optic nerve, which forms a large communication bottleneck between the retina and the lateral geniculate nucleus. But the brain is not just a set of carefully formatted codes winging their way back and forth through cleverly constructed communication channels. Rather, it is a computing device. Within its tangled tendrils latent variables are constructed, manipulated, combined; hypotheses evaluated; models built; risks evaluated; actions selected; limbs coordinated. When peeking inside such a great machine we should expect to find the fumes of the ongoing processing, more than the formatting of messages. Hence the major motivation of this thesis: normative theories for neurons computing, not communicating.

There are various computations the nervous system is performing. This thesis aims to develop mathematical tools that can help us reason about how a circuit might perform a given computation, and how that computation will manifest in neural activity. In Marr's three levels these represent the algorithmic and implementational level. Such tools let us link our most common observations, neural activity, to objects of deep interest, the ongoing computations, in ways that permit prediction, control, and inference - guessing the computation from the measured neural activity. This leaves us with two obvious questions: which computations? And how will we construct a mathematical theory to think about their neural instantiations?

In this part of the thesis the answer to the first question will be internal modelling: building internal circuitry that reflects some structure in the world around us. In particular, given states and actions we ask neural circuitry to implement a forward model: predict new state from old state and action. To answer the second question we will assume two things. First, a particular form of internal model: that the transformations of the variable being modelled form a group[1]. Second, following evidence from well-understood neural circuitry, we'll assume that the way the circuit updates its encoding is an action-dependent linear map. Together, this will allow us to use the maths of group representation theory to constrain the potential neural implementations, and reason about their optimal encoding.

World Models The world contains endlessly repeating structures at every possible level of inquiry: water quality predicts disease, gravitational ripples reflect merging black holes, rivers predictably swell with spring meltwater and shrivel under the summer sun. In every direction that humanity looks we have constructed some structure for understanding that domain's intricacies of cause and effect. We act like scientists - to variable degrees of accuracy. The reason why we do this is clear: models permit prediction. Knowing the consequences of your actions is an enormous boon for survival. It allows you to plan your actions by simulating internally rather than learning from risky real-world trial-and-error. Or, if you understand the world's logical rules, you can make inferences: normally unsupported objects either fall or are birds, therefore that levitating metal room is probably an alien spaceship. Indeed, it is likely only because the world is replete with arcane structures that intelligence was a useful thing to evolve.

Neurally Implementing a World Model What might a neural implementation of such a world model look like? A simple formalisation would be via a set of states, s, and actions, a: given a current state and action the model has to predict what the next state is going to be, $s' = f_a(s)$, a classic control theory forward model. We'll consider limitations of this model later, but this framework encapsulates finite state machine or physical model, making it sufficiently flexible to be interesting. How can we construct such a system using neural circuitry?

[1]Which turns out to be convenient but not completely necessary, relaxations are possible

One classic approach is exemplified by the cerebellum, and analogous structures found throughout nervous systems. Here, both state and action are encoded as input neural activity, i.e. the firing of cerebellar mossy fibres $\boldsymbol{m}(s, a)$, then simple non-linear feed-forward processing is used to calculate the predicted state $\boldsymbol{o}(s') = h(\boldsymbol{m}(s, a))$ (Cerminara, Apps, and Marple-Horvat, 2009; R. Miall, 2007). This approach has a lot of benefits. One-hidden layer neural networks like the cerebellum are universal function approximators given sufficient hidden-layer neurons (Cybenko, 1989), and the hidden layer of the cerebellum, the granule cells, comprise roughly half the neurons in the human brain, making it able to learn many models. Further, there are relatively well understood biologically plausible learning algorithms that operate on the hidden-output (granule-purkinjie) connections, making the learning of arbitrary mappings not just mathematically possible, but biologically plausible[2]. Given these attractive properties it is perhaps not surprising that all vertebrates have a cerebellum (Nieuwenhuys, Ten Donkelaar, and Nicholson, 2014), and analogous structures, such as the mushroom body, are used by invertebrates to solve related problems (Aso et al., 2014; Strausfeld et al., 1998).

However, this approach has an obvious limitation. It is well structured to swill together the outputs of your sensors with the efferent copies from your motor system to construct predictions of your future. It is less good at constructing, tracking, and storing meaningful latent variables that you can use to understand the world. For this, some kind of recurrence is needed to stabilise the representation of a variable in isolation from ongoing sensory or motor input. These considerations lead us to our second option, more reflective of neocortex: recurrent circuitry that can maintain an encoding of state, $\boldsymbol{g}(s)$, which is then updated upon arrival of an action signal, $\boldsymbol{u}(a)$, via some recurrent update h, to encode the new state $\boldsymbol{g}(s')$:

$$\boldsymbol{g}(s') = h(\boldsymbol{g}(s), \boldsymbol{u}(a)) \tag{2.1}$$

Through correctly tuned recurrent circuitry, this representation can be limitlessly maintained, a form of memory.

These two modeling systems, recurrent and feedforward, cortical and cerebellar, can feed off one another. Once you have some circuitry that is able to maintain a latent variable, it can also form an input and a prediction of a feedforward cerebellar internal model. This allows greater flexibility, with recurrent circuitry storing and updating meaningful latent variables, and feedforward circuitry learning arbitrary update functions.

Therefore, we shall study the recurrent form of internal model, in particular, the neural evidence for the existence and structuring of these circuits across nervous systems.

Recurrent World Models & Continuous Attractor Networks The most well-studied set of internal models concern navigational variables, such as an animal's heading direction, self-position, or homing vector. These are clearly useful, and can be updated by integrating various parts of your velocities (variables that are not so far removed from vestibular signals, motor efferent copy, or sensory patterns such as visual flow). The circuit implementation of these internal models form one of the most beautiful stories in neuroscience, and one of the biggest success stories of computational/theoretical neuroscience.

Neurons encoding heading direction were first discovered in rats (Taube, 2007; Taube, R. U. Muller, and Ranck, 1990a,b) but have since been found in mice (Yoder and Taube, 2009), fish (Petrucco et al., 2023), flies (Seelig and Jayaraman, 2015), cockroaches (Varga and Ritzmann, 2016), and butterflies (Beetz et al., 2022) Following the original discovery, computational models were developed to answer the two basic questions: what has to be true about a neural circuit such that it (1) stably encodes an angle over a long period of time, (2) can appropriately update its encoding when the animal turns (Ben-Yishai, Bar-Or, and Sompolinsky, 1995; Redish, Elga, and Touretzky, 1996; Skaggs et al., 1994; X. Xie, Hahnloser, and Seung, 2002). In recent years the circuitry first hypothesised in the 90s has proven an astonishingly good match to heading direction circuits in the insect central complex (Hulse and Jayaraman, 2020), and the two dimensional analogue of this circuit appears to underly the grid cell self-position system (Vollan et al., 2025). As such, the principles underlying this circuitry are the obvious target for reasoning about biologically plausible models of recurrent circuitry.

So how do these circuits work? Take the simplest case, encoding heading direction, a single angle. To stably encode the angle we will use one population of neurons, the compass neurons, each of which will end up being tuned to a single direction. We can therefore think of this population as being arrayed on a ring according to their tuning direction[3]. Most simply, we can choose the recurrent circuitry so that neurons that are nearby on the ring strongly excite one another, while they inhibit those on the other side of the ring. Various versions of this connectivity that respect Dale's law can also be used, for example local excitation amongst compass neurons with global inhibition from a persistently active inhibitory neuron. This connectivity exhibits a translational symmetry, the connectivity between two neurons at positions θ_i and θ_j on the ring only depends on the distance between the angles: $W(\theta_i, \theta_j) = W(|\theta_i - \theta_j|)$. If this local excitatory connectivity is strong enough then the stable patterns of activity in this network are localised bumps: one set of neighbouring cells are active, all others are silent. It is then possible to use the position of this bump as a stable encoding of the heading direction.

[2]Since learning appears to happen mostly on output synapses, cerebellar-networks more resemble kernel regression than they do end-to-end trained neural networks, a link we'll exploit in part IV

[3]this ordering is purely a theoretical/functional construct, the neurons don't actually have to live in a ring, yet in the insect central complex they do (Seelig and Jayaraman, 2015) - exactly as wire minimisation would suggest!

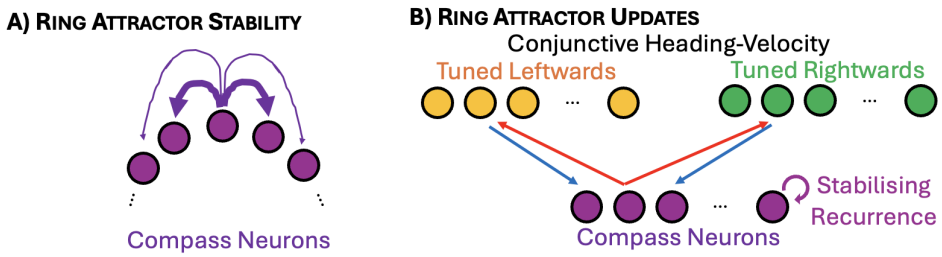


Figure 2.1: A: Local recurrent connectivity amoungst compass neurons stabilises an encoding of a single bump of activity, encoding a heading direction. B: Shifter circuits, populations of conjunctive heading-velocity neurons with shifted connectivity, can implement the appropriate bump movements in the circuit to track a changing heading angle.

Next we need to make the bump move when a velocity signal arrives, this can done with a set of 'shifter neurons' (Anderson and Van Essen, 1987). Let's focus on turning the bump one direction, clockwise. To do this we create one shifter population, each neuron in this population is paired with a compass neuron and receives excitatory input from that neuron and, to a lesser degree, its neighbours. It then projects back into the ring with the same connectivity pattern but shifted one neuron clockwise. When this neuron is active it therefore upweights the activity one step clockwise, which encourages that shifted angle to win the winner-takes-all dynamics, pushing the bump around. How do we specify the activity of this neuron to ensure it pushes the bump correctly? Crucially this neuron is connected not only to the compass, but also receives an angular velocity input when the animal is turning clockwise. It then activates in a conjunction: the animal both has to be turning clockwise and the bump has to be close to this neuron's peak tuning for the neuron to activate. If both those conditions are satisfied the shifter neuron will be highly active and the bump will move around. Finally, the functional dependence of the conjunctive neuron can be chosen such that the speed of the bump's movement is proportional to the angular velocity input.

The full circuit therefore comprises one set of compass neurons, the 'state encoding neurons' in our internal model framing, and two sets of shifter neurons, one tuned to clockwise movements, the other to anticlockwise. Our description was purposefully simplistic, much work has gone into studying these circuits and showing they are reasonably robust. Further, this circuitry, with some elaborations, matches precisely to the circuitry of the insect central complex. It is a huge vote of confidence in neuroscience that models dreamed up in the 90s on the basis of first-principles considerations have proved so similar to real neural networks.

Gated Linear Network Models We would like to generalise these ideas, and analyse how such a system should optimally be constructed. To do this we will introduce some simplifications that permit mathematical analysis. In particular, we follow analogous arguments to those Logiaco, Abbott, and Escola (2021) used in modelling thalamo-cortical loops. We begin with the state encoding or compass neurons, $\boldsymbol{g}(t)$. These are recurrently connected with weights $\boldsymbol{W}$, and each group of shifter neurons $\boldsymbol{t}_a(t)$ is connected to the state neurons via a connectivity matrix $\boldsymbol{B}_a$. We model these dynamics as a linear dynamical system:

$$\frac{d\boldsymbol{g}}{dt} = -\boldsymbol{g}(t) + \boldsymbol{W}\boldsymbol{g}(t) + \sum_a \boldsymbol{B}_a \boldsymbol{t}_a(t) \tag{2.2}$$

We conceptualise each group of shifter neurons as being effectively gated by their action: the clockwise neurons are on when the animal is turning clockwise, or off otherwise. Further, we assume they are not recurrently connected amongst themselves (as seems to be observed), but that they do receive input from the state neurons through connectivity matrices $\boldsymbol{A}_a$. When gated on, we assume the dynamics of the shifter neurons are also linear:

$$\frac{d\boldsymbol{t}_a}{dt} = -\boldsymbol{t}_a + \boldsymbol{A}_a \boldsymbol{g}(t) \quad \text{if action} = a \qquad \text{else} \quad \boldsymbol{t}_a(t) = \boldsymbol{0} \tag{2.3}$$

Since these dynamics have no recurrence we assume they settle to equilibrium ($\boldsymbol{t}_a = \boldsymbol{A}_a \boldsymbol{g}(t)$) before the state neurons appreciably change. This allows us to write the state neuron dynamics as:

$$\frac{d\boldsymbol{g}}{dt} = \boldsymbol{W}\boldsymbol{g}(t) + \underbrace{\boldsymbol{B}_a \boldsymbol{t}_a(t)}_{\text{gated on shifters}} = (\boldsymbol{W} + \boldsymbol{B}_a \boldsymbol{A}_a)\boldsymbol{g}(t) \tag{2.4}$$

i.e. the dynamics evolves under an action-dependent linear dynamical system. In this case we used discrete actions so the gating implements the action coding. However we can extend to continuous actions in various ways.

The mathematically simplest is to assume that the equilibrium point of the shifter neurons is multiplicatively scaled by the velocity of that action, $v_a(t)$:

$$\boldsymbol{t}_a = v_a(t)\boldsymbol{A}_a\boldsymbol{g}(t) \tag{2.5}$$

This introduces a multiplicative scaling in the recurrent dynamics:

$$\frac{d\boldsymbol{g}}{dt} = (\boldsymbol{W} + v_a(t)\boldsymbol{B}_a\boldsymbol{A}_a)\boldsymbol{g}(t) \tag{2.6}$$

and we can still understand the dynamics as forming some kind of action dependent linear dynamical system. This set of assumptions might seem somewhat arbitrary. One justification is simply that it ensures tractability while still capturing the key phenomena, as we shall see. Alternatively, some of my future work attempts to argue there is more to it than that: it seems that task-optimised neural networks might be biased towards finding these kinds of solutions. For now, we'll take at face-value the idea that we can model the effect of updates in this circuitry via an action-dependent linear map and use it to frame a precise optimisation problem.

Formalised: The Efficient Computing Hypothesis Now we frame a representational optimisation problem for the encoding of an internal model. This is a set of objectives and constraints on the state encoding, $\boldsymbol{g}(s)$ that fall into three natural categories. The first two are simply instantiations of the efficient coding hypothesis:

> FUNCTIONAL: To be a good encoding of state, different states have to be encoded differently. There are many ways of operationalising this. You could, for example, ask for a linear readout to be able to decode the state. Or you could avoid specifying a decoder and ask for a high mutual information between the code and the variable. We will specify this as the problem demands.
>
> BIOLOGICAL: You can enforce various efficiency or biological constraints. We will always constrain the firing rates to be non-negative, and either penalise or constrain the norm of the firing rate vectors.

However, an optimisation problem like this has no notion of internal modelling, the key thing we are trying to capture! We therefore introduce our third constraint, motivated by our previous modeling discussion, which contains the only novelty in the problem:

> ACTIONABLE: To be an internal model we require that an action dependent linear map is able to update the representation:
>
> $$\boldsymbol{g}(s' = s + a) = \boldsymbol{W}(a)\boldsymbol{g}(s) \tag{2.7}$$
>
> If a representation satisfies this constraint, i.e. a set of $\boldsymbol{W}(a)$ matrices can be constructed that implement the required update, then we could construct a plausible neural circuit implementation using the linear dynamical system formulation above.

Hence we get our internal modeling optimisation problem: under the constraints of actionability, be a good efficient coding representation. This is the precise form of the Efficient Computing Hypothesis, the conjecture that a good model of neural firing is that it is efficient, subject to the constraints inherent in performing the required computation.

Links to Group and Representation Theory Now we have turned the idea 'be an internal model' into a very simple equation, eq. (2.7). How does such a set of equations constrain the neural representation? In other words, within this framework, what does it mean for neural activity to 'be an internal model'?

Fortunately in many cases a simple twist on well-established mathematics is enough to answer these questions. If the set of actions forms a mathematical object called a group then group representation theory tells us precisely the implied constraints. Let's take the example of movement on a ring. The structure of the space enforces constraints on the matrices. For example, in order to be consistent the matrix that rotates you by $\Delta\theta$ must be the inverse of the matrix that rotates you by $-\Delta\theta$ in order that:

$$\forall\theta \quad \boldsymbol{g}(\theta) = \boldsymbol{g}(\theta + \Delta\theta - \Delta\theta) = \boldsymbol{W}(\Delta\theta)\boldsymbol{W}(-\Delta\theta)\boldsymbol{g}(\theta) = \boldsymbol{g}(\theta) \qquad \boldsymbol{W}(\Delta\theta) = \boldsymbol{W}(-\Delta\theta)^{-1} \tag{2.8}$$

Similarly $\boldsymbol{W}(2\Delta\theta) = \boldsymbol{W}(\Delta\theta)^2$, and rotation by 2π corresponds to no movement: $\boldsymbol{W}(2\pi) = \mathbb{1}$. In other words, the structure of the space being modeled induces constraints on the set of matrices that can implement these actions: $\{\boldsymbol{W}(\Delta\theta)\forall\Delta\theta \in [0, 2\pi]\}$. Group representation theory is the area of maths that exactly characterises how these constraints manifest in the sets of allowable matrices, which will be very useful.

It turns out that each group possesses a (potentially infinite) set of what are called irreducible representation, a set of matrices that implement all the rules correctly, but cannot be decomposed into simpler systems that do. For example the simple rotation matrix is a irreducible representation of the group of rotations:

$$\boldsymbol{R}(\Delta\theta) = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \tag{2.9}$$

It obeys all the rules, $\boldsymbol{R}(2\Delta\theta) = \boldsymbol{R}(\Delta\theta)^2$, $\boldsymbol{R}(2\pi) = \mathbb{1}$, etc. But that is not the only option, you could rotate twice as fast and all the rules still hold:

$$\boldsymbol{R}_2(\Delta\theta) = \begin{bmatrix} \cos(2\Delta\theta) & -\sin(2\Delta\theta) \\ \sin(2\Delta\theta) & \cos(2\Delta\theta) \end{bmatrix} \tag{2.10}$$

In fact, rotating at any integer frequency will satisfy all the rules of the group, including not rotating at all, an option called the trivial irreducible rotation:

$$\boldsymbol{R}_0(\Delta\theta) = \begin{bmatrix} 1 \end{bmatrix} \tag{2.11}$$

You can see these irreducible representations, or irreps to use the lingo, can have different dimensionalities. Stacking them and invertibly mixing them produces another matrix that obeys all the rules:

$$\boldsymbol{W}(\Delta\theta) = \boldsymbol{A} \begin{bmatrix} \boldsymbol{R}_1(\Delta\theta) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_2(\Delta\theta) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \boldsymbol{A}^{-1} \tag{2.12}$$

But $\boldsymbol{W}(\Delta\theta)$, while a representation of the group, is not an irrediculble representation of a group, since it can be broken down into parts which are themselves representations of the group.

It turns out that every representation of the group is made from an invertible linear mixture of irreps of the group. Further, for the 1D rotation group we've been describing (i.e. rotations on the ring), we've already described every irrep. Therefore we can write every action matrix as:

$$\boldsymbol{W}(\Delta\theta) = \boldsymbol{A} \begin{bmatrix} \boldsymbol{R}_{\omega_1}(\Delta\theta) & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{R}_{\omega_D}(\Delta\theta) \end{bmatrix} \boldsymbol{A}^{-1} \tag{2.13}$$

For some set of irreps indexed by their frequencies: $\{\omega_d\}_{d=1}^D$. As long as the sum of the dimensionalities of each of the irreps is less than the dimensionality of the whole space (the number of neurons) this is a feasible set of action matrices. These were all statements about the action matrices, but they can easily be turned into statements about the neural code since:

$$\boldsymbol{g}(\theta) = \boldsymbol{W}(\theta)\boldsymbol{g}(0) = \boldsymbol{A} \begin{bmatrix} \boldsymbol{R}_{\omega_1}(\Delta\theta) & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{R}_{\omega_D}(\Delta\theta) \end{bmatrix} \boldsymbol{A}^{-1}\boldsymbol{g}(0) \tag{2.14}$$

Hence the neural activity is just a linear combination of these frequencies:

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \cos(\omega_d\theta) + \boldsymbol{b}_d \sin(\omega_d\theta) \tag{2.15}$$

Where $D \leq \lfloor \frac{N}{2} \rfloor$, with some constraints on the set of coefficient vectors $\{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^D$ such that (1) if sine a one frequency is in the code, so is its cosine (so that the whole irrep is present) and (2) the vectors are linearly independent such that the matrix $\boldsymbol{A}$ is invertible.

This is now a precise constraint that means 'be an internal model': to be an actionable internal model of a periodic 1D variable the code must be made from a small number of frequencies. From this point we can proceed as before, optimising the coefficients in the code $\{\boldsymbol{a}_d, \boldsymbol{b}_d, \omega_d\}_{d=1}^D$ to form an efficient encoding of the relevant variables, but we can do this knowing that this representation can form the substrate for the computations we are interested in. It is not just a passive encoding, it is a representation with which the system can compute.

Relationship to other models The mathematical formulation of a set of context-specific linear system controlling the state often recurs. This is promising, since it means there are a lot of tools available.

- **Grid Cells** This framework has appeared in neuroscience before, even in the modelling of grid cells (Gao, J. Xie, Wei, et al., 2021; Gao, J. Xie, Zhu, et al., 2019; Issa and K. Zhang, 2012; J. Whittington et al., 2018; J. C. Whittington, T. H. Muller, et al., 2020; Yilun Xu et al., 2020). These works all showed that, using various architectures and losses, a code respecting these constraints can be optimised to look like grid cells. That said, none of these works framed the problem in the same way as we have here, nor did they make use of group representation theory to analyse the resulting 'path-integrating' constraint, nor could they fit the multi-modular structure. We delay extensive comparisons to these works to our grid cell theory review, chapter 4.

- **Switching Linear Systems** These are dynamical models in which at each timepoint the dynamics are linear, but which linear system is operating can change through time. They have been successfully used to model neural data (Linderman, Johnson, et al., 2017; Linderman, A. Nichols, et al., 2019; Nassar et al., 2018). Our system is similarly a switching linear system, with the switches governed by the action signal. This correspondence suggests tools for learning these models from data.

- **Linear Relational Embedding** Another work sought to model concepts as vectors and relationships as linear maps between them, then aimed to learn generalisations from a limited set of concept-relation-concept triples (for example, if Annette is Christopher's mother, and Philip is Christopher's brother, then we know that Annette is also Christopher's mother) (Paccanaro and G. E. Hinton, 2000, 2001, 2002). This suggests models like this have interesting generalisation properties, that we won't study further here.

- **Billinear Models** These seek to fit data that is a nonlinear function of two variables as a billinear combination (Tenenbaum and Freeman, 2000). A datapoint vector $\boldsymbol{y}_t$ is modelled as $y_{ti} = \sum_{jk} W_{ijk} a_{tj} b_{tk}$ where $\boldsymbol{a}_t$ and $\boldsymbol{b}_t$ represent the two variables, and W_{ijk} are their combinations. Performing one of the sums leads us to a linear map of one variable that is dependent on another, as in our framing: $y_{ti} = \sum_k (\sum_j W_{ijk} a_{tj}) b_{tk}$, $\boldsymbol{y}_t = \boldsymbol{W}(\boldsymbol{a}_t)\boldsymbol{b}_t$.

- **Feature-wise Linear Mapping** FiLM learns to adapt a neural network output on the basis of a contextual features (Perez et al., 2018). The adaption is an affine function dependent on the context, just like our context dependent linear map.

Conclusions and Future Work We have tried to motivate our problem setting. We began with the concept of internal models in neural systems, and settled on studying their recurrent implementations as is in RNNs, or as hypothesised in cortex. We then introduced an approximation: action-gated linear systems, that has proven effective before, and is a natural choice made in a series of other settings. We analyse this choice using group representation theory and find we are able to make a very concrete encapsulation of the internal modelling constraint, perfect for future analysis in a more standard efficient coding theory. There are many ways this kind of idea is limited: when should we expect our assumption to work? How would a neural system learn such a representation? And what should we do when the actions don't form a group? These questions won't be answered in this thesis, though we'll return to them in the concluding remarks. Despite this, the next chapter will shows us that using these ideas we can get a long way studying grid cells chapter 3, and in part III we will similarly apply these ideas to understanding working memory representations in prefrontal cortex.

Chapter 3

Actionable Neural Representations

To afford flexible behaviour, the brain must build internal representations that mirror the structure of variables in the external world. For example, 2D space obeys rules: the same set of actions combine in the same way everywhere (step north, then south, and you won't have moved, wherever you start). We suggest the brain must represent this consistent meaning of actions across space, as it allows you to find new short-cuts and navigate in unfamiliar settings. We term this representation an 'actionable representation'. We formulate actionable representations using group and representation theory, and show that, when combined with biological and functional constraints - non-negative firing, bounded neural activity, and precise coding - multiple modules of hexagonal grid cells are the optimal representation of 2D space. We support this claim with intuition, analytic justification, and simulations. Our analytic results normatively explain a set of surprising grid cell phenomena, and make testable predictions for future experiments. Lastly, we highlight the generality of our approach beyond just understanding 2D space. Our work characterises a new principle for understanding and designing flexible internal representations: they should be actionable, allowing animals and machines to predict the consequences of their actions, rather than just encode.

3.1 Introduction

Animals should build representations that afford flexible behaviours. However, different representation make some tasks easy and others hard; representing red versus white is good for understanding wines but less good for opening screw-top versus corked bottles. A central mystery in neuroscience is the relationship between tasks and their optimal representations. Resolving this requires understanding the representational principles that permit flexible behaviours such as zero-shot inference.

Here, we introduce **actionable representations**, a representation that permits flexible behaviours. Being actionable means encoding not only variables of interest, but also how the variable transforms. Actions cause many variables to transform in predictable ways. For example, actions in 2D space obey rules; north, east, south, and west, have a universal meaning, and combine in the same way everywhere. Embedding these rules into a representation of self-position permits deep inferences: having stepped north, then east, then south, an agent can infer that stepping west will lead home, having never taken that path - a zero-shot inference (Figure 3.1A).

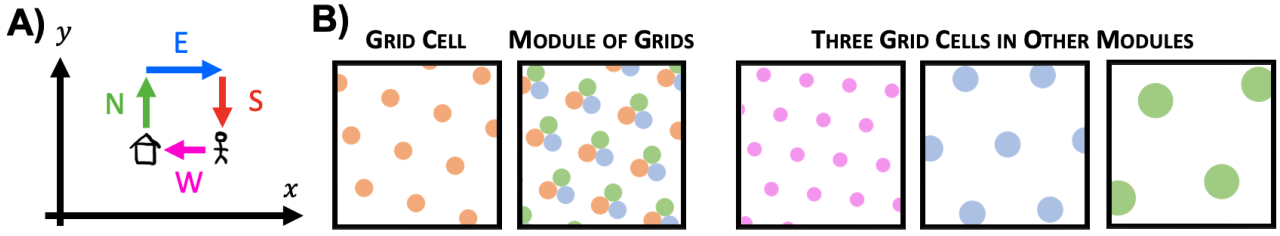


Figure 3.1: **A** 2D space is defined by rules, e.g. at all positions north = −south. **B** Left: Entorhinal grid cells are hexagonally tuned cells (orange). Different cells within a module are translated copies (orange/blue/green). Right: Different modules have different lattice scale (pink/blue/green).

Indeed biology represents 2D space in a structured manner. Grid cells in medial entorhinal cortex represent an abstracted 'cognitive map' of 2D space (Tolman, 1948). These cells fire in a hexagonal lattice of positions (Hafting et al., 2005), (Figure 3.1B), and are organised in modules; cells within one module have receptive fields

that are translated versions of one another, and different modules have firing lattices of different scales and orientations (Figure 3.1B), (H. Stensola et al., 2012).

Biological representations must be more than just actionable - they must be **functional**, encoding the world efficiently, and obey **biological** constraints. We formalise these three ideas - actionable, functional, and biological - and analyse the resulting optimal representations. We define *actionability* using group and representation theory, as the requirement that each action has a corresponding matrix that linearly updates the representation; for example, the 'step north' matrix updates the representation to its value one step north. *Functionally*, we want different points in space to be represented maximally differently, allowing inputs to be distinguished from one another. *Biologically*, we ensure all neurons have non-negative and bounded activity. From this constrained optimisation problem we derive optimal representations that resemble multiple modules of grid cells.

Our problem formulation allows analytic explanations for grid cell phenomena, matches experimental findings, such as the alignment of grids cells to room geometry (T. Stensola et al., 2015), and predicts some underappreciated aspects, such as the relative angle between modules. In sum, we 1) propose actionable neural representations to support flexible behaviours; 2) formalise the actionable constraint with group and representation theory; 3) mix actionability with biological and functional constraints to create a constrained optimisation problem; 4) analyse this problem and show that in 2D the optimal representation is a good model of grid cells, thus offering a mathematical understanding of why grid cells look the way they do; 5) provide several neural predictions; 6) highlight the generality of this normative method beyond 2D space.

3.2 Actionable Neural Representations: An Objective

We seek a representation $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^N$ of 2D position $\boldsymbol{x} \in \mathbb{T}^2$, where N is the number of neurons (formally $\boldsymbol{x}$ lives on the torus, discussed further below). Our representation is built using three ideas: functional, biological, and actionable; whose combination will lead to multiple modules of grid cells, and which we'll now formalise.

Functional: To be useful, the representation must encode different positions differently. However, it is more important to distinguish positions 1km apart than 1mm, and frequently visited positions should be separated the most. To account for these, we ask our representation to minimise

$$\mathcal{L} = \iint e^{-\frac{\|\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{g}(\boldsymbol{x}')\|^2}{2\sigma^2}} \chi(\boldsymbol{x},\boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' \tag{3.1}$$

The red term measures the representational similarity of $\boldsymbol{x}$ and $\boldsymbol{x}'$; it is large if their representations are nearer than some distance σ in neural space and small otherwise. By integrating over all pairs $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\mathcal{L}$ measures the total representational similarity, which we seek to minimise. The green term is the agent's position occupancy distribution, which ensures only visited points contribute to the loss, for now simply a Gaussian of lengthscale L. Finally, the blue term weights the importance of separating each pair, encouraging separation of points more distant than a lengthscale, l.

$$\chi(\boldsymbol{x},\boldsymbol{x}') = 1 - e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2l^2}} \qquad p(\boldsymbol{x}) = e^{-\frac{\|\boldsymbol{x}\|^2}{2L^2}} \tag{3.2}$$

Biological: Neurons have non-negative firing rates, so we constrain $\boldsymbol{g}_i(\boldsymbol{x}) \geq 0$. Further, neurons can't fire arbitrarily fast, and firing is energetically costly, so we constrain each neuron's response $g_n(\boldsymbol{x})$ via $\int g_n^2(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = 1$

Actionable: Our final constraint requires that the representation is actionable. This means each transformations of the input must have its own transformation in neural space, independent of position. For mathematical convenience we enact the neural transformation using a matrix. Labelling this matrix $\boldsymbol{T}(\Delta\boldsymbol{x}) \in \mathbb{R}^{N\times N}$, for transformation $\Delta\boldsymbol{x}$, this means that for all positions $\boldsymbol{x}$,

$$\boldsymbol{g}(\boldsymbol{x}+\Delta\boldsymbol{x}) = \boldsymbol{T}(\Delta\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x}) \tag{3.3}$$

For intuition into how this constrains the neural code $\boldsymbol{g}(\boldsymbol{x})$, we consider a simple example of two neurons representing an angle $\theta \in [0, 2\pi)$. Replacing $\boldsymbol{x}$ with θ in eq. (3.3) we get the equivalent constraint: $\boldsymbol{g}(\theta+\Delta\theta) = \boldsymbol{T}(\Delta\theta)\boldsymbol{g}(\theta)$. Here the matrix $\boldsymbol{T}$ performs a rotation, and the solution (up to a linear transform) is for $\boldsymbol{T}$ to be the standard 2×2 rotation matrix, with frequency n.

$$\boldsymbol{g}(\theta+\Delta\theta) = \begin{pmatrix}\cos(n[\theta+\Delta\theta])\\ \sin(n[\theta+\Delta\theta])\end{pmatrix} = \begin{pmatrix}\cos(n\Delta\theta) & -\sin(n\Delta\theta)\\ \sin(n\Delta\theta) & \cos(n\Delta\theta)\end{pmatrix}\begin{pmatrix}\cos(n\theta)\\ \sin(n\theta)\end{pmatrix} = \boldsymbol{T}(\Delta\theta)\boldsymbol{g}(\theta) \tag{3.4}$$

Thus as θ rotates by 2π the neural representation traces out a circle an integer number, n, times. Thanks to the problem's linearity, extending to more neurons is easy. Adding two more neurons lets the population contain

another sine and cosine at some frequency, just like the two neurons in equation 3.4. Extrapolating this we get our actionability constraint: the neural response must be constructed from some invertible linear mixing of the sines and cosines of $D < \frac{N}{2}$ frequencies,

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(n_d\theta) + \boldsymbol{b}_d \cos(n_d\theta) \qquad \text{for integer } n_d \tag{3.5}$$

The vectors $\{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^{D} \in \mathbb{R}^N$ are coefficient vectors that mix together the sines and cosines, of which there are D. $\boldsymbol{a}_0$ is the coefficient vector for a frequency that cycles 0 times.

This argument comes from an area of maths called Representation Theory (a different meaning of representation!) that places constraints on the matrices $\boldsymbol{T}$ for variables whose transformations form a mathematical object called a group. This includes many of interest, such as position on a circle, torus, or sphere. These constraints on matrices can be translated into constraints on an actionable neural code just like we did for $\boldsymbol{g}(\theta)$ (see Appendix 3.A). When generalising the above example to 2D space (a torus), we must consider a few things: First, the space is two-dimensional, so compared to our previous equation 3.5, the frequencies, denoted $\boldsymbol{k}_d$, are now two dimensional. Second, to approximate a finite region of flat 2D space, we consider a similarly sized region of a torus. As the radius of the torus grows this approximation becomes arbitrarily good (see Appendix 3.A.4 for discussion). Periodicity constrains the frequencies in equation 3.5 to be $\frac{n}{R}$ for integer n and ring radius R. As the loop (torus in 2D) becomes very large these permitted frequencies become arbitrarily close, so we drop the integer constraint,

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(\boldsymbol{k}_d \cdot \boldsymbol{x}) + \boldsymbol{b}_d \cos(\boldsymbol{k}_d \cdot \boldsymbol{x}) \tag{3.6}$$

Our constrained optimisation problem is complete. Equation 3.6 specifies the set of actionable representations. Without additional constraints these codes are meaningless: random combinations of sines and cosines produce random neural responses (Figure 3.2A). We will choose from amongst the set of actionable codes by optimising the parameters $\boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d, \boldsymbol{k}_d\}_{d=1}^{D}$ to minimise $\mathcal{L}$, subject to biological (non-negative and bounded firing rates) constraints.

3.3 Optimal Representations

Optimising over the set of actionable codes to minimise $\mathcal{L}$ with biological constraints gives multiple modules of grid cells (Figure 3.2B). This section will, using intuition and analytics, explain why.

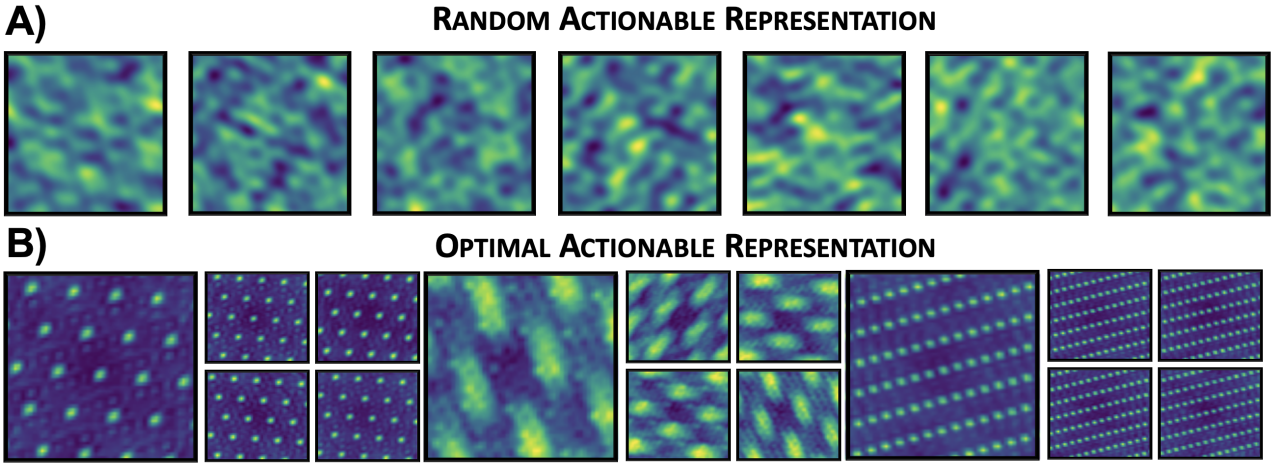


Figure 3.2: **A** Random actionable representations (equation 3.6) are meaningless combinations of sines and cosines. ($g_n(\boldsymbol{x})$ plotted for different neurons, n) **B** Optimising among actionable codes to achieve functional and biological constraints produces multiple modules of ∼hexagonal grid cells. (Figure 3.6 shows that all the neurons in the population belong to one of these three modules)

3.3.1 Non-negativity Leads to a Module of Lattice Cells

To understand how non-negativity produces modules of lattice responses we will study the following simplified loss, which maximises the *Euclidean distance* between representations of angle, $\boldsymbol{g}(\theta)$,

$$\mathcal{L}_0 = -\frac{1}{4\pi^2}\iint_{-\pi}^{\pi} \|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2 d\theta d\theta' \tag{3.7}$$

This is equivalent to the full loss (equation 3.1) for uniform $p(\theta)$, $\chi(\theta, \theta') = 1$, and σ very large. Make no mistake, this is a bad loss. For contrast, the full loss encouraged the representations of different positions to be separated by more than σ, enabling discrimination[1]. Therefore, sensibly, the representation is most rewarded for separating nearby (closer than σ) points. $\mathcal{L}_0$ does the opposite! It grows quadratically with separation, so $\boldsymbol{g}(\theta)$ is most rewarded for pushing apart already well-separated points, a terrible representational principle! Nonetheless, $\mathcal{L}_0$ will give us key insights.

Since actionability gives us a parameterised form of the representations (equation 3.5), we can compute the integrals to obtain the following constrained optimisation problem (details: Appendix 3.C)

$$\min_{\substack{\boldsymbol{a}_0,\\ \{\boldsymbol{a}_d,\boldsymbol{b}_d,n_d\}_{d=1}^D}} \mathcal{L}_0 = -\sum_{d=1}^{D} \|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2 \quad \text{with} \quad \overbrace{\boldsymbol{g}(\theta) > 0,}^{\text{Non-negativity}} \quad \overbrace{\|\boldsymbol{a}_0\|^2 + \frac{1}{2}\sum_{d=1}^{D} \|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2 = N}^{\text{Bounded firing rates}} \tag{3.8}$$

Where N is the number of neurons. This is now something we can understand. First, reminding ourselves that the neural code, $\boldsymbol{g}(\theta)$, is made from a constant vector, $\boldsymbol{a}_0$, and $\theta-$dependent parts (equation 3.5; Figure 3.3A), we can see that $\mathcal{L}_0$ separates representations by encouraging the size of each varying part, $\|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2$, to be maximised. This effect is limited by the firing rate bound, $\|\boldsymbol{a}_0\|^2 - \frac{1}{2}\mathcal{L}_0 = N$. Thus, to minimise $\mathcal{L}_0$ we must minimise the constant vector, $\boldsymbol{a}_0$. This would be easy without non-negativity (when any code with $\|\boldsymbol{a}_0\| = 0$ is optimal), but no sum of sines and cosines can be non-negative for all θ without an offset. Thus the game is simple; choose frequencies and coefficients so the firing rates are non-negative, but using the smallest possible constant vector.

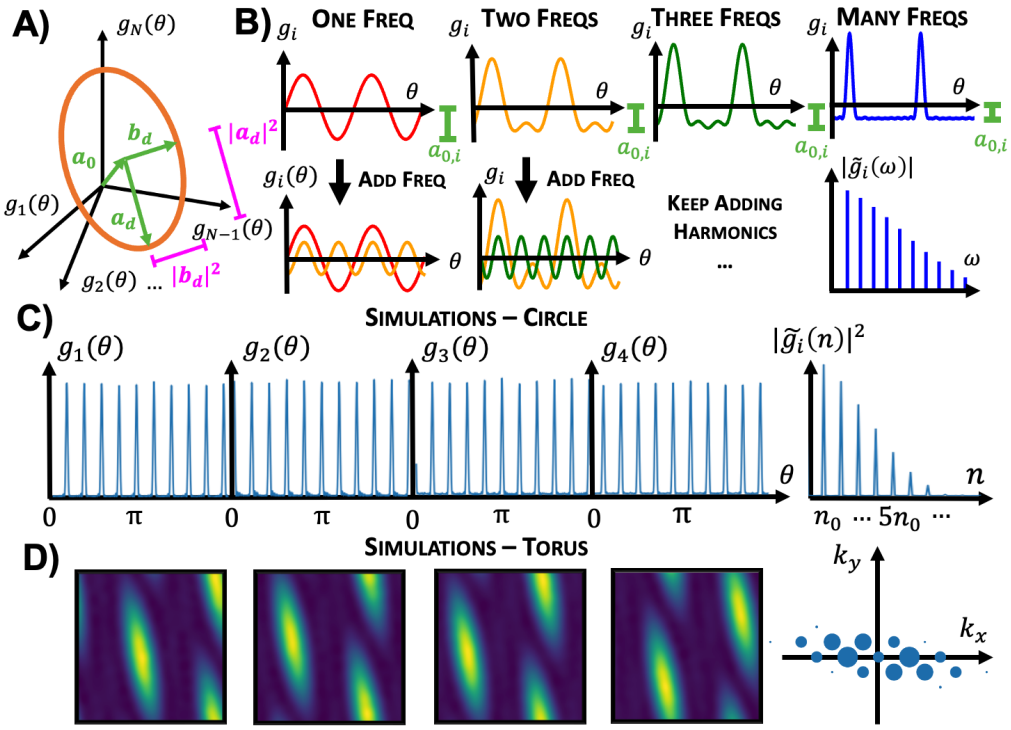


Figure 3.3: **A** The neural activity consists of a constant vector, $\boldsymbol{a}_0$, and $\theta-$dependent loops. **B** Progressively adding harmonic frequencies increases the code's minima, allowing the code to be made non-negative using the smallest possible $\boldsymbol{a}_0$. This give a grid-like tuning curve. Simulations results confirm this heuristic in **C** 1D and **D** 2D, right: frequency spectrum, 2D dot size is frequency power.

[1] σ could be interpreted as a noise level, or a minimum discriminable distance, then points should be far enough away for a downstream decoder to distinguish them.

One lattice cell. We now heuristically argue, and confirm in simulation, that the optimal solution for a single neuron is a lattice tuning curve Starting with a single frequency component, e.g. $\sin(\theta)$, achieving non-negativity requires adding a constant offset, $\sin(\theta) + 1$ (Figure 3.3B). However, we could also have just added another frequency. In particular adding harmonics of the base frequency (with appropriate phase shifts) pushes up the minima (Figure 3.3B). Extending this argument, we suggest non-negativity, for a single cell, can be achieved by including a grid of frequencies. This gives a lattice tuning curve (Figure 3.3B right).

Module of lattice cells. Achieving non-negativity for this cell used up many frequencies. But as discussed (Section 3.2), actionability only allows a limited number frequencies in the population ($< \frac{N}{2}$ since each frequency uses 2 neurons (sine and cosine)), thus how can we make lots of neurons non-negative with limited frequencies? Fortunately, we can do so by making all neuron's tuning curves translated versions of each other, as translated curves contain the same frequencies but with different phases. This is a module of lattice cells. We validate our arguments by numerically optimising the coefficients $\boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^D$ and frequencies $\{n_d\}_{d=1}^D$ to minimise $\mathcal{L}_0$ subject to constraints, producing a module of lattices (Figure 3.3C; details in Appendix 3.B). These arguments equally apply to representations of a periodic 2D space (a torus; Figure 3.3D).

Studying $\mathcal{L}_0$ has told us why lattice response curves are good. But surprisingly, all lattices are equally good, even at infinitely high frequency. Returning to the full loss will break this degeneracy.

3.3.2 Prioritising Important Pairs of Positions Produces Hexagonal Grid Cells

Now we return to the full loss and understand its impact in two steps, beginning with the reintroduction of χ and p, which break the lattice degeneracy, forming hexagonal grid cells.

$$\mathcal{L} = \iint_{-\infty}^{\infty} e^{-\frac{\|\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{g}(\boldsymbol{x}')\|^2}{2\sigma^2}} \chi(\boldsymbol{x}, \boldsymbol{x}') p(\boldsymbol{x}) p(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \tag{3.9}$$

χ prefers low frequencies: recall that $\chi = 1 - e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2l^2}}$ ensures very distant inputs have different representations, while allowing similar inputs to have similar representations, up to a resolution, l. This encourages low frequencies ($\|\boldsymbol{k}_d\| < \frac{1}{l}$), which separate distant points but produce similar representations for pairs closer than l (Analytics: Appendix 3.D.1). At this stage, for periodic 2D space, the lowest frequency lattices, place cells, are optimal (see Appendix 4; Sengupta et al., 2018).

$p(\boldsymbol{x})$ prefers high frequencies: However, the occupancy distribution of the animal, $p(\boldsymbol{x})$, counters χ. On an infinite 2D plane animals must focus on representing a limited area, of lengthscale L. This encourages high frequencies ($\|\boldsymbol{k}_d\| > \frac{1}{L}$), whose response varies among the visited points (Analytics: Appendix 3.D.2). More complex $p(x)$ induce more complex frequency biases, but, to first order, the effect is always a high frequency bias (fig. 3.5F-G, Appendix 3.I).

Combination → Hexagons: Satisfying non-negativity and functionality required a lattice of many frequencies, but now p and χ bias our frequency choice, preferring those beyond $\frac{1}{L}$ (to separate points the animal visits) but smaller than $\frac{1}{l}$ (to separate distant visited points). Thus to get as many of these preferred frequencies as possible, we want the lattice with the densest packing within a Goldilocks annulus in frequency space (Figure 3.4A). This is a hexagonal lattice in frequency space which leads to a hexagonal grid cell. Simulations with few neurons agree, giving a module of hexagonal grid cells (Figure 3.4B).

3.3.3 A Harmonic Tussle Produces Multiple Modules

Finally, we will study the neural lengthscale σ, and understand how it produces multiple modules.

$$\mathcal{L} = \iint_{-\infty}^{\infty} e^{-\frac{\|\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{g}(\boldsymbol{x}')\|^2}{2\sigma^2}} \chi(\boldsymbol{x}, \boldsymbol{x}') p(\boldsymbol{x}) p(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \tag{3.10}$$

As discussed, $\mathcal{L}$ prioritises the separation of poorly distinguished points, those whose representations are closer than σ. This causes certain frequencies to be desired *in the overall population*, in particular those unrelated to existing frequencies by simple harmonic ratios, i.e. not $\omega_1 = \frac{3}{2}\omega_2$ (Figure 3.4C; see Appendix 3.E for a perturbative derivation of this effect). This is because pairs of harmonically related frequencies represent more positions identically than a non-harmonically related pair, so are worse for separation (similar to arguments made in Wei, Prentice, and Balasubramanian, 2015).

This, however, sets up a 'harmonic tussle' between what the population wants - non-harmonically related frequencies for $\mathcal{L}$ - and what single neurons want - harmonically related frequency lattices for non-negativity (Section 3.3.1). Modules of grid cells resolve this tension: harmonic frequencies exist within modules to give non-negativity, and non-harmonically related modules allow for separation, explaining the earlier simulation results (Figure 3.2B; further details in Appendix 3.E.3).

This concludes our main result. We have shown three constraints on neural populations - actionable, functional, and biological - lead to multiple modules of hexagonal grid cells, and we have understood why. We

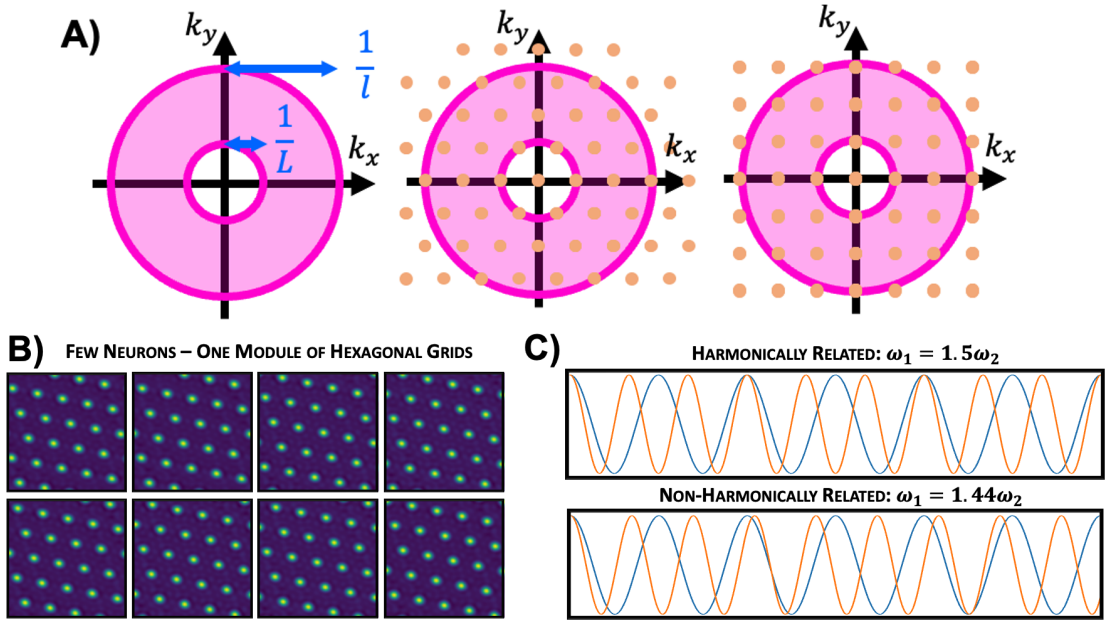


Figure 3.4: **A** χ and $p(\boldsymbol{x})$ induce a bias towards frequencies with magnitudes between $\frac{1}{L}$ and $\frac{1}{l}$. Since, of all lattices, hexagons fit the most frequencies within the annulus, they are preferred, and hexagonal frequency lattices lead to hexagonal grid cells. **B** Simulations confirm. **C** Harmonically related frequencies co-repeat more often than non-harmonic, meaning that, as a pair, harmonically related frequencies are worse at encoding, since they encode many points in the same way.

posit this is the minimal set of requirements for grid cells (see Appendix 3.F for ablations simulations and discussion).

3.4 Predictions

Our theory makes testable predictions about the structure of optimal actionable codes for 2D space. We describe three here: tuning curve sharpness scales with the number of neurons in a module; the optimal angle between modules; and the optimal grid alignment to room geometry.

3.4.1 Lattice Size:Field Width Ratio scales with Number of Neurons in Module

In our framework the number of neurons controls the number of frequencies in the representation (equation 3.6). A neuron within a module only contains frequencies from that module's frequency lattice, since other modules have non-harmonically related frequencies. More neurons in a module, means more and higher frequencies in the lattice, which sharpen grid peaks, fig. 3.5A. We formalise this (Appendix 3.G) and predict that the number of neurons within a module scales with the square of the lattice lengthscale, ν, to field width, μ, ratio, $N \propto \left(\frac{\nu}{\mu}\right)^2$. This matches the intuition that the sharper a module's peak, the more neurons you need to tile the entire space. In a rudimentary analysis, our predictions compare favourably to data from H. Stensola et al., 2012 assuming uniform sampling of grid cells across modules, fig. 3.5B. We are eager to test these claims quantitatively.

3.4.2 Modules are Optimally Oriented at Small Offsets ($\sim 4°$)

In section 3.3.3 we saw how frequencies of different modules are maximally non-harmonically related in order to separate the representation of as many points as possible. To maximise non-harmonicity between two modules, the second module's frequency lattice can be both stretched *and* rotated relative to the first. 0 or 30° relative orientations are particularly bad coding choices as they align the high density axes of the two lattices, fig. 3.5C. The optimal angular offset of two modules, calculated via a frequency overlap metric (Appendix 3.H), is small, fig. 3.5D; the value depends on the grid peak and lattice lengthscales, μ and ν, but varies between 3° and 8° degrees. Multiple modules should orient at a sequence of small angles (Appendix 3.H). In a rudimentary analysis, our predictions compare favourably to the observations of H. Stensola et al., 2012, fig. 3.5E.

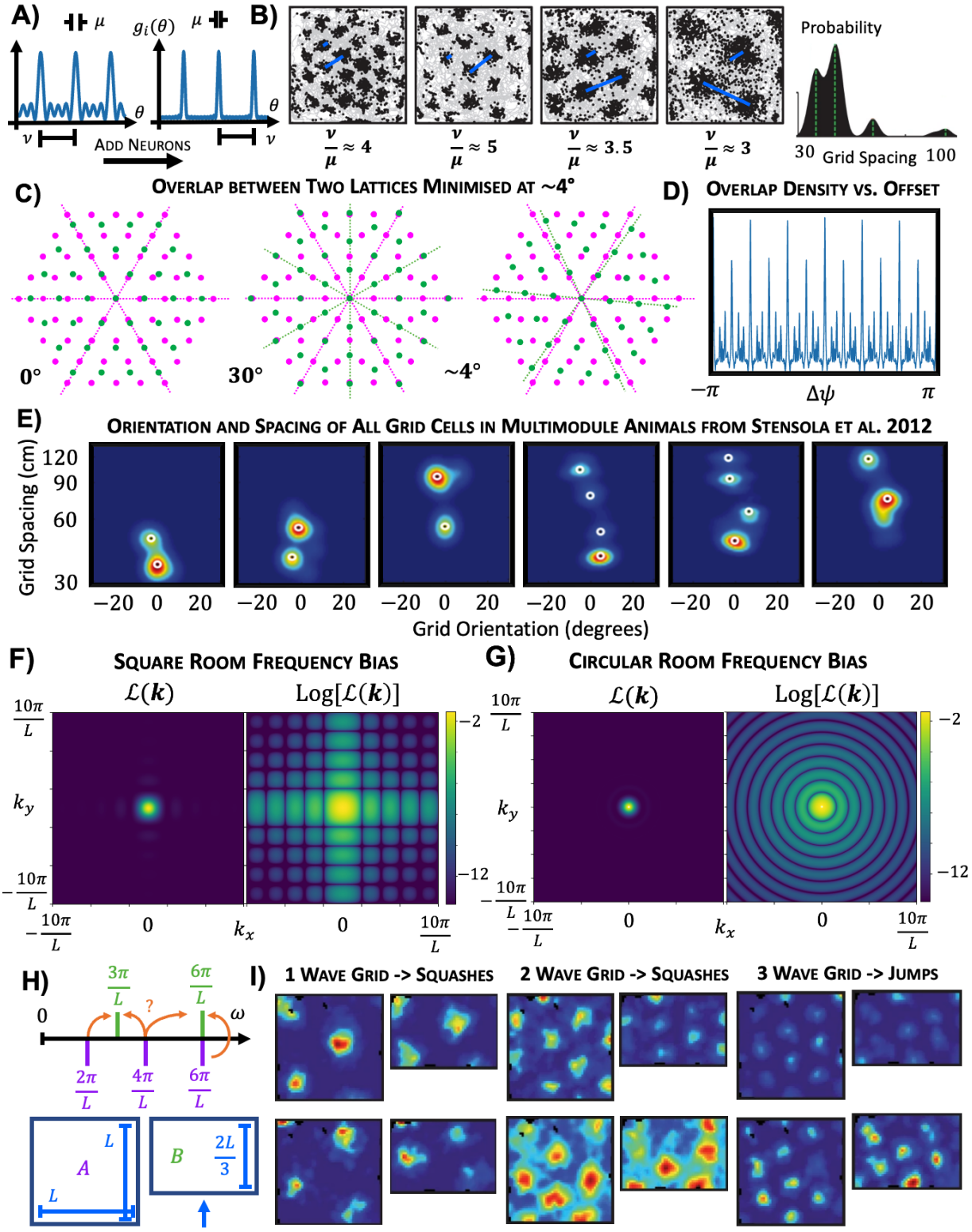


Figure 3.5: **A** Adding neurons to a module sharpens the grid peaks. **B** In data from H. Stensola et al., 2012 the most sharply peaked grids were recorded most often (2nd from left), and the broadest the least (rightmost). **C** Aligning the high-density axes of two lattices creates high overlap, while small offsets minimise it. **D** We quantified the overlap as a function of offset angle (Appendix 3.H) and found the minima occured at $\sim 4°$ from aligned (aligned = 6 maxima, 4° offsets are the 12 minima). **E** The orientation and spacing of all grids in animals with multiple modules recorded by H. Stensola et al., 2012. Many modules are misaligned by small offsets, matching our prediction. **F** Square or **G** circular rooms create complex frequency biases, $\mathcal{L}(\boldsymbol{k})$ is the loss for a one frequency code of fixed amplitude. **H** The optimal frequencies along one axis of a box occur at $\frac{2\pi n}{L}$ for integer n: Squishing a room makes the optimal frequencies expand. Grids should change to fit the optimal patterns in the recorded environment, unless they happen to be optimal for both, as $\frac{6\pi}{L}$ is. **I** Most grid cells scale with the room, but, when one side is squashed by a factor of 2/3, those at $\frac{6\pi}{L}$ are stable.

3.4.3 Optimal Grids Morph to Room Geometry

In Section 3.3.2 (and Appendix 3.D) we showed that $p(x)$, the animal's occupancy distribution, introduced a high frequency bias - grid cells must peak often enough to encode visited points. However, changing $p(x)$ changes the shape of this high frequency bias (Appendix 3.I). In particular, we examine an animal's encoding of square, circular, or rectangular environments, Appendix 3.I, with the assumption that $p(x)$ is uniform over that space. In each case the bias is coarsely towards high frequencies, but has additional intricacies: in square and rectangular rooms optimal frequencies lie on a lattice, with peaks at integer multiples of $\frac{2\pi}{L}$ along one of the cardinal axes, for room width/height L, fig. 3.5F; whereas in circular rooms optima are at the zeros of a Bessel function, fig. 3.5G. These ideas make several predictions. For example, grid modules in circular rooms should have lengthscales set by the optimal radii in fig. 3.5G, but they should still remain hexagonal since the Bessel function is circularly symmetric. However, the optimal representation in square rooms should not be perfectly hexagonal since $p(x)$ induces a bias inconsistent with a hexagonal lattice (this effect is negligible for high frequency grids). Intriguingly, shearing towards squarer lattices is observed in square rooms (T. Stensola et al., 2015), and it would be interesting to test its grid-size dependence.

Lastly, these effects make predictions about how grid cells morph when the environment geometry changes. A grid cell that is optimal in both geometries can remain the same, however sub-optimal grid cells should change. For example turning a square into a squashed square (i.e. a rectangle), stretches the optimal frequencies along the squashed dimension. Thus, some cells are optimal in both rooms and should stay stable, will others will change, presumably to nearby optimal frequencies (Figure 3.5H). Indeed H. Stensola et al., 2012 recorded the same grid cells in a square and rectangular environment (Figure 3.5I), and observed exactly these phenomena.

3.5 Discussion & Conclusions

We have proposed actionability as a fundamental representational principle to afford flexible behaviours. We have shown in simulation and with analytic justification that the optimal actionable representations of 2D space are, when constrained to be both biological and functional, multiple modules of hexagonal grid cells, thus offering a mathematical understanding of grid cells. We then used this theory to make three novel grid cell predictions that match data on early inspection.

While this is promising for our theory, there remain some grid cell phenomena that, as it stands, it will never predict. For example, grid cell peaks vary in intensity (Dunn et al., 2017), and grid lattices bend in trapezoidal environments (Krupic et al., 2015). These effects may be due to incorporation of sensory information or uncertainty - things we have not included - to better infer position. Including these may recapitulate these findings, similar to Ocko et al., 2018 and Y. H. Kang, Wolpert, and Lengyel, 2023.

Our theory is normative and abstracted from implementation. However, both modelling (Burak and I. R. Fiete, 2009) and experimental (R. J. Gardner et al., 2022; Kim et al., 2017) work suggests that continuous attractor networks (CANs) implement path integrating circuits. In chapter 2 we saw that our actionability condition is an approximation of exactly these kinds of circuits.

While we focused on understanding the optimal representations of 2D space and their relationship to grid cells, our theory is more general. Most simply, it can be applied to behaviour in other, non 2D, spaces. In fact many variables whose transformations form a group are relatively easily analysed. The brain represents many such variables, e.g. heading directions, (Finkelstein et al., 2015), object orientations, (Logothetis, Pauls, and Poggio, 1995), the'4-loop task' of Sun et al., 2020 or 3-dimensional space (Ginosar et al., 2021; Grieves et al., 2021). Interestingly, our theory predicts 3D representations with regular order (Figure 3.19E in Appendix 3.J), unlike those found in the brain (Ginosar et al., 2021; Grieves et al., 2021) suggesting the animal's 3D navigation is sub-optimal.

Further, the brain represents these differently-structured variables not one at a time, but simultaneously; at times mixing these variables into a common representation (Hardcastle et al., 2017), at others giving each variable its own set of neurons (e.g. grid cells, object-vector cells Høydal et al., 2019). Thus, one potential concern about our work is that it assumes a separate neural population represents each variable. In the next part of this thesis, part II, we answer this concern, describing principles that justify the existence of meaningful categories of cells, such as grid cells. Further, in chapter 6, we apply these results specifically to grid cells, more comprehensively justifying the existence of multiple modules of grid cells.

But, most expansively, these principles express a view that representations must be more than just passive encodings of the world; they must embed the consequences of predictable actions, allowing planning and inferences in never-before-seen situations. We codified these ideas using Group and Representation theory, and demonstrated their utility in understanding grid cells. However, the underlying principle is broader than the crystalline nature of group structures: the world and your actions within it have endlessly repeating structures whose understanding permits creative analogising and flexible behaviours. A well-designed representation should reflect this.

Technical Appendices: Actionable Neural Representations

3.A Constraining Representations with Representation Theory

Having an actionable code means the representational effect of every transformation of the variable, $\Delta\boldsymbol{x}$, can be implemented by a matrix:

$$\boldsymbol{g}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{T}(\Delta\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x}) \tag{3.11}$$

In Section 3.2 we outlined a rough argument for how this equation leads to the following constraint on actionable representations of 2D position, that was vital for our work:

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(\boldsymbol{k}_d \cdot \boldsymbol{x}) + \boldsymbol{b}_d \cos(\boldsymbol{k}_d \cdot \boldsymbol{x}) \tag{3.12}$$

In this section, we'll repeat this argument more robustly using Group and Representation Theory. In doing so, it'll become clear how broadly our version of actionability can be used to derive clean analytic representational constraints; namely, the arguments presented here can be applied to the representation of any variable whose transformations form a group whose representations are well understood.

Sections 3.A.1 and 3.A.2 contain a review of the Group and Representation theory used. Section 3.A.3 applies it to our problem.

3.A.1 Group Theory

A mathematical group is a collection of things (like the set of integers), and a way to combine two members of the group that makes a third (like addition of two integers, that always creates another integer), in a way that satisfies a few rules:

1. There is an identity element, which is a member of the group that when combined with any other element doesn't change it. For adding integers the identity element is 0, since $a + 0 = a$ for all a.
2. Every member of the group has an inverse, defined by its property that combining an element with its inverse produces the identity element. In our integer-addition example the inverse of any integer a is $-a$, since $-a + a = 0$, and 0 is the identity element.
3. Associativity applies, which just means the order in which you perform operations doesn't matter: $(a + b) + c = a + (b + c)$

Groups are ubiquitous. We mention them here because they will be our route to understanding actionable representations - representations in which transformations are also encoded consistently. The set of transformations of many variables of interest are groups. For example, the set of transformations of 2D position, i.e. 2D translations - $\Delta\boldsymbol{x}$, is a group if you define the combination of two translations via simple vector addition, $\Delta\boldsymbol{x}_{1+2} = \Delta\boldsymbol{x}_1 + \Delta\boldsymbol{x}_2$. We can easily check that they satisfy all three of the group rules:

1. There is an identity translation: simply add $\mathbf{0}$.
2. Each 2D translation has its inverse: $-\Delta\boldsymbol{x} + \Delta\boldsymbol{x} = \mathbf{0}$
3. Associativity: $(\Delta\boldsymbol{a} + \Delta\boldsymbol{b}) + \Delta\boldsymbol{c} = \Delta\boldsymbol{a} + (\Delta\boldsymbol{b} + \Delta\boldsymbol{c})$

The same applies for the transformations of an angle, θ, or of positions on a torus or sphere, and much else besides. This is a nice observation, but in order to put it to work we need a second area of mathematics, Representation Theory.

3.A.2 Representation Theory

Groups are abstract, in the sense that the behaviour of many different mathematical objects can embody the same group. For example, the integers modulo P with an addition combination rule form a group, called C_P. But equally, the P roots of unity ($\{e^{\frac{2\pi n}{P}}\}_{n=0}^{P-1}$) with a multiplication combination rule obey all the same rules: you can create a 1-1 correspondence between the integers 0 through $P-1$ and the P roots of unity by labelling all the roots with the integer n that appears in the exponent $e^{\frac{2\pi n}{P}}$, and, under their respective combination rules, all additions of integers will exactly match onto multiplication of complex roots (i.e. $e^{\frac{2\pi n_1}{P}} * e^{\frac{2\pi n_2}{P}} = e^{\frac{2\pi \bmod P(n_1+n_2)}{P}}$)

In this work we'll be interested in a particular instantiation of our groups, defined by sets of matrices, $\boldsymbol{T}(\Delta\boldsymbol{x})$, combined using matrix multiplication. For example, a matrix version of the group C_P would be the set of 2-by-2 rotation matrices that rotate by $\frac{2\pi}{P}$ increments:

$$\boldsymbol{T}(n) = \begin{pmatrix} \cos(\frac{2\pi n}{P}) & -\sin(\frac{2\pi n}{P}) \\ \sin(\frac{2\pi n}{P}) & \cos(\frac{2\pi n}{P}) \end{pmatrix} \qquad \text{such that} \quad \boldsymbol{T}(n_1)\boldsymbol{T}(n_2) = \boldsymbol{T}(\operatorname{mod}_P(n_1+n_2)) \tag{3.13}$$

Combining these matrices using matrix multiplication follows all the same patterns that adding the integers modulo P, or multiplying the P roots of unity, followed; they all embody the same group.

Representation theory is the branch of mathematics that specifies the structure of sets of matrices that, when combined using matrix multiplication, follow the rules of a particular group. Such sets of matrices are called a representation of the group; to avoid confusion arising from this unfortunate though understandable convergent terminology, we distinguish group representations from neural representations by henceforth denoting *representations* of a group in italics. We will make use of one big result from *Representation* Theory, hinted at in Section 3.2: the Peter-Weyl theorem (Knapp, 2002). For compact topological groups (which include transformations of a point on a circle, torus, and sphere) any matrix that is a *representation* of a group can be composed from the direct product of a set of blocks, called irreducible *representations*, or *irreps* for short, up to a linear transformation. i.e. if $\boldsymbol{T}(\Delta\boldsymbol{a})$ is a *representation* of the group of transformations of some variable $\boldsymbol{a}$:

$$\boldsymbol{T}(\Delta\boldsymbol{a}) = \boldsymbol{S}\begin{bmatrix} I_1(\Delta\boldsymbol{a}) & 0 & 0 & \dots & 0 \\ 0 & I_2(\Delta\boldsymbol{a}) & 0 & \dots & 0 \\ 0 & 0 & I_3(\Delta\boldsymbol{a}) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I_D(\Delta\boldsymbol{a}) \end{bmatrix}\boldsymbol{S}^{-1} \tag{3.14}$$

where $I_d(\Delta\boldsymbol{a})$ are the *irreps* of the group in question, which are square matrices not necessarily of the same size as d varies, and $\boldsymbol{S}$ is some invertible square matrix.

To motivate our current rabbit hole further, this is exactly the result we hinted towards in Section 3.2. We discussed $\boldsymbol{T}(\Delta\theta)$, and, in 2-dimensions, argued it was, up to a linear transform, the 2-dimensional rotation matrix. Further, we discussed how every extra two neurons allowed you to add another frequency, i.e. another rotation matrix. This was an attempt to motivate the plausibility of the Peter-Weyl theorem: the rotation matrices are the *irreps* of the group of transformations of an angle, and adding two neurons allows you to create a larger $\boldsymbol{T}(\Delta\theta)$ by stacking rotation matrices on top of one another. Now including the invertible linear transform, $\boldsymbol{S}$, we can state the 4-dimensional version of equation 3.4:

$$\boldsymbol{T}(\Delta\theta) = \boldsymbol{S}\begin{pmatrix} \cos(n_1\Delta\theta) & -\sin(n_1\Delta\theta) & 0 & 0 \\ \sin(n_1\Delta\theta) & \cos(n_1\Delta\theta) & 0 & 0 \\ 0 & 0 & \cos(n_2\Delta\theta) & -\sin(n_2\Delta\theta) \\ 0 & 0 & \sin(n_2\Delta\theta) & \cos(n_2\Delta\theta) \end{pmatrix}\boldsymbol{S}^{-1} \tag{3.15}$$

In performing this decomposition the role of the rotation matrices as *irreps* is clear. There are infinitely many types, each specified by an integer frequency, n_d, and their direct product produces any *representation* of the rotation group.

We can now return to actionable neural representations of periodic 2D space, a torus, where this theorem comes in very useful. We sought codes, $\boldsymbol{g}(\boldsymbol{x})$, that could be manipulated via matrices:

$$\boldsymbol{g}(\boldsymbol{x}+\Delta\boldsymbol{x}) = \boldsymbol{T}(\Delta\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x}) \tag{3.16}$$

Now we recognise that we're asking $\boldsymbol{T}(\Delta\boldsymbol{x})$ to be a *representation* of the transformation group of $\boldsymbol{x}$, and we know the shape $\boldsymbol{T}(\Delta\boldsymbol{x})$ must take to fit this criteria. To specify $\boldsymbol{T}(\Delta\boldsymbol{x})$ from this constrained class we must simply choose a set of *irreps* that fill up the dimensionality of the neural space, stack them one on top of each other, and rotate and scale them using the matrix $\boldsymbol{S}$.

A final detail is that there is a trivial way to make a *representation* of any group: choose $\boldsymbol{T}(\theta) = \mathbb{I}_N$, the N-by-N identity matrix. This also fits our previous discussion, but corresponds to a representation made from the direct product of N copies of something called the trivial *irrep*, a 1-dimensional *irrep*, $I_{\text{trivial}}(\Delta\boldsymbol{a}) = 1$.

Armed with this knowledge, the following subsection will show how this theory can be used to constrain the neural code, $\boldsymbol{g}(\boldsymbol{x})$. To finish this review subsection, we list the non-trivial *irreps* of the groups used in this work. To be specific, in all our work we use only the real-valued *irreps*, i.e all elements of $\boldsymbol{T}$ are real numbers, since firing rates are real numbers. These are less common than complex *irreps*, which is often what is implied by the simple name *irreps*.

Transformation group of which variable	Non-trivial Real *Irreps*
An angle on a unit circle, θ	$\begin{pmatrix}\cos(n\Delta\theta) & -\sin(n\Delta\theta)\\ \sin(n\Delta\theta) & \cos(n\Delta\theta)\end{pmatrix}$ for $n \in \mathbb{Z}$
Position on a very big circle $\approx$ a line, x	$\begin{pmatrix}\cos(\omega\Delta\theta) & -\sin(\omega\Delta\theta)\\ \sin(\omega\Delta\theta) & \cos(\omega\Delta\theta)\end{pmatrix}$ for $\omega \in \mathbb{R}$
Angles on a unit torus, $\boldsymbol{\theta}$	$\begin{pmatrix}\cos(\boldsymbol{a}\cdot\Delta\boldsymbol{\theta}) & -\sin(\boldsymbol{a}\cdot\Delta\boldsymbol{\theta})\\ \sin(\boldsymbol{a}\cdot\Delta\boldsymbol{\theta}) & \cos(\boldsymbol{a}\cdot\Delta\boldsymbol{\theta})\end{pmatrix}$ for $\boldsymbol{a} \in \mathbb{Z}^2$
Position on a very big torus $\approx$ plane, $\boldsymbol{x}$	$\begin{pmatrix}\cos(\boldsymbol{k}\cdot\Delta\boldsymbol{x}) & -\sin(\boldsymbol{k}\cdot\Delta\boldsymbol{x})\\ \sin(\boldsymbol{k}\cdot\Delta\boldsymbol{x}) & \cos(\boldsymbol{k}\cdot\Delta\boldsymbol{x})\end{pmatrix}$ for $\boldsymbol{k} \in \mathbb{R}^2$
Position on a unit sphere, ϕ	Real Wigner-D Matrices

Proofs and discussions for deriving these *irreps* of the circle, torus, and sphere transformation groups can be found in any textbook on the *representation* theory of compact Lie Groups (e.g. Knapp (2002)). The step from complex to real *irreps* can be done using the Frobenius-Schur indicator, which gives a recipe for mapping from the complex *irreps* of a compact group to the real *irreps* (Fulton and J. Harris, 1991). The real Wigner D-Matrices were calculated recursively as detailed in Ivanic and Ruedenberg (1996, 1998), by translating matlab code from Politis et al. (2016) into python.

Our derivations of the *irreps* on the very large circle and torus are simple. In 1D the constraint that the frequencies n be integers is only because θ lives on the unit circle. Then the frequencies are constrained such that after rotating by 2π the function is identical. If you change the radius of the circle to R this constraint becomes $\omega_d = \frac{n_d}{R}$ where n_d is any integer. As you take the circle's radius to infinity, in the process making finite sections of the circle a better and better approximation of finite patches of flat 1D space, the lattice of permitted frequencies ω_d becomes arbitrarily close together. Eventually they are separated by machine precision, and so we can simply implement the code as if they were continuous, hence $\omega_d \in \mathbb{R}$. The same argument applies analogously for 2D frequencies on a very very large torus.

3.A.3 Representational Constraints

Finally, we will translate these constraints on transformation matrices into constraints on the neural code. This is, thankfully, not too difficult. Consider the representation of an angle for simplicity, and take some arbitrary origin in the input space, $\theta_0 = 0$. The representation of all other angles can be derived via:

$$\boldsymbol{g}(\theta) = \boldsymbol{T}(\theta)\boldsymbol{g}(0) = \boldsymbol{S}\begin{bmatrix} I_1(\theta) & 0 & 0 & \dots & 0 \\ 0 & I_2(\theta) & 0 & \dots & 0 \\ 0 & 0 & I_3(\theta) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I_D(\theta) \end{bmatrix}\boldsymbol{S}^{-1}\boldsymbol{g}(0) \tag{3.17}$$

In the case of an angle, each *irrep*, $I_d(\theta)$, is just a 2-by-2 rotation matrix at frequency d, section 3.A.2, and for an N-dimensional $\boldsymbol{T}$ we can fit maximally $\frac{N}{2}$ (for N even) different frequencies, where N is the number of neurons. Hence the representation, $\boldsymbol{g}(\theta)$, is just a linear combination of the sine and cosine of these different frequencies, exactly as quoted previously:

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(n_d\theta) + \boldsymbol{b}_d \cos(n_d\theta) \qquad \text{for integer } n_d, \tag{3.18}$$

$\boldsymbol{a}_0$ corresponds to the trivial *irrep*, and we include it since we know it must be in the code to make the firing rates non-negative for all θ. It also cleans up the constraint on the number of allowable frequencies, for even or odd N we require $D < \frac{N}{2}$. This is because if N is odd one dimension is used for the trivial *irrep*, the rest can be shared up amongst $\frac{N-1}{2}$ frequencies, so D must be an integer smaller than $\frac{N}{2}$. If N is even, one dimension must still be used by the trivial *irrep*, so D can still maximally be only the largest integer smaller than $\frac{N}{2}$.

Extending this to other groups is relatively simple, the representation is just a linear combination of the functions that appear in the *irrep* of the group in question, see section 3.A.2. For position on a circle, line (very large circle), torus, or plane (very large torus), they all take the relatively simple form as in equation 3.18, but

requiring appropriate changes from 1 to 2 dimensions, or integer to real frequencies. Representations of position on a sphere are slightly more complex, instead being constructed from linear combinations of sets of spherical harmonics.

3.A.4 Periodic vs Infinite Spaces

We finally give further details about a key step in our argument for representations of flat 2D space. We approximate a finite region of the infinite 2D plane by a finite region of a very large periodic 2D space, the torus. We do this because the *representation* theory of the group of 2D translations is not as well characterised (to the best our knowledge there is no equivalent of the Peter-Weyl theorem that could be used to provide a target for optimisation). Fortunately, any animal only cares about a finite region of 2D space (encoded in equation 3.2 via the lengthscale L). We therefore approximate this finite region of flat 2D space with an equivalently sized region of a torus. This enables us to use the fully characterised *representation* theory of the set of translations of periodic space.

Now, there are legitimate concerns over whether this is a reasonable approximation. Fortunately, we are free to choose the radii of the torus as we wish, and we make use of this freedom to make the approximation of the finite flat 2D space arbitrarily good. As the radii increase to infinity the small region of torus approximates the flat 2D space arbitrarily well.

This use of periodic space does exclude some representations of the full set of 2D translations, for example:

$$\boldsymbol{T}(\Delta\boldsymbol{x}) = \boldsymbol{S}\begin{pmatrix}1 & 0 & \Delta x\\ 0 & 1 & \Delta y\\ 0 & 0 & 1\end{pmatrix}\boldsymbol{S}^{-1} \tag{3.19}$$

There are two reasons not to be concerned by this. First, this solution would never have been allowed, as including it ensures that your representation, $\boldsymbol{g}(\boldsymbol{x})$, cannot have non-negative or bounded firing rates (we suspect this would be the case for all such additional representations of flat 2D translations). Second, any result which was dependent on these kind of boundary effects at ∞ would be highly suspect. After all, animals are not truly representing flat 2D space, rather they are representing an approximately flat section of the surface of a sphere (the Earth).

3.B Numerical Optimisation Details

In order to test our claims, we numerically optimise the parameters that define the representation to minimise the loss, subject to constraints. Despite sharing many commonalities, our optimisation procedures are different depending on whether the variable being represented lives in a very large periodic space (approximations to the line, plane, or volume) or finite (circle, torus, sphere). We describe each of these schemes in turn, beginning with the very large spaces. All code is available on github: `https://github.com/WilburDoz/ICLR_Actionable_Reps`.

3.B.1 Numerical Optimisation for Very Large Spaces

We will use the representation of a point on a line for explanation, planes or volumes are a simple extension. $\boldsymbol{g}(x)$ is parameterised as follows:

$$\boldsymbol{g}(x) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \cos(\omega_d x) + \boldsymbol{b}_d \sin(\omega_d x) \qquad \boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^{D} \in \mathbb{R}^N, \omega_d \in \mathbb{R} \tag{3.20}$$

Our loss is made from three terms: the first is the functional objective, the last two enforce the non-negativity and boundedness constraint respectively:

$$\mathcal{L} = \mathcal{L}_{\text{functional}} + \lambda_{\text{p}} \mathcal{L}_{\text{non-negativity}} + \lambda_{\text{b}} \mathcal{L}_{\text{bounded}} \tag{3.21}$$

To evaluate the functional component of our loss we sample a set of points $\{x_i\}_{i=1}^{M}$ from $p(x)$ and calculate their representations $\{\boldsymbol{g}(x_i)\}_{i=1}^{M}$. To make the bounded constraint approximately satisfied (there's still some work to be done, taken care of by $\mathcal{L}_{\text{bounded}}$) we calculate the following neuron-wise norm and use it to normalise each neuron's firing:

$$\|g_n\|^2 = \sum_{m=1}^{M} g_n(x_m)^2 \qquad \tilde{g}_n(\theta) = \frac{g_n(\theta)}{\|g_n\|} \tag{3.22}$$

The functional form of $\mathcal{L}_{\text{functional}}$ varies, as discussed in the main paper. For example, for the full loss we compute the following using the normalised firing, a discrete approximation to equation 3.1:

$$\mathcal{L}_{\text{functional}} = \frac{1}{M^2} \sum_{m=1}^{M} \sum_{m'=1}^{M} e^{-\frac{\|\tilde{\boldsymbol{g}}(x_m) - \tilde{\boldsymbol{g}}(x_{m'})\|^2}{2\sigma^2}} \chi(x_m, x_{m'}) \tag{3.23}$$

Now we come to our two constraints, which enforce that the representation is non-negative and bounded. We would like our representation to be reasonable (i.e. non-negative and bounded) for all values of x. If we do not enforce this then the optimiser finds various trick solutions in which the representation is non-negative and bounded only in small region, but negative and growing explosively outside of this, which is completely unbiological, and uninteresting. Of course, x is infinite, so we cannot numerically ensure these constraints are satisfied for all x. However, ensuring they are true in a region local to the animal's explorations (which are represented by $p(x)$) suffices to remove most of these trick solutions. As such, we sample a second set of M_S 'shift positions', $\{x_m\}_{m=1}^{M_S}$, from a scaled up version of $p(x)$, using a scale factor S. We then create a much larger set of positions, by shifting the original set by each of the 'shift positions', creating a dataset of size $M * M_S$, and use these to calculate our two constraint losses.

Our non-negativity loss penalises negative firing rates:

$$\mathcal{L}_{\text{non-negativity}} = \frac{1}{M M_S N} \sum_{m_s=1}^{M_S} \sum_{m=1}^{M} \sum_{n=1}^{N} |\tilde{g}_n(x_{m_s} + x_m)| \mathbb{I}(\tilde{g}_n(x_{m_s} + x_m) < 0) \tag{3.24}$$

Where $\mathbb{I}$ is the indicator function, 1 if the firing rate is negative, 0 else, i.e. we just average the magnitude of the negative portion of the firing rates.

The bounded loss penalises the deviation of each neuron's norm from 1, in each of the shifted rooms:

$$\mathcal{L}_{\text{bounded}} = \frac{1}{N M_S} \sum_{n=1}^{N} \sum_{m_s=1}^{M_S} (\sum_{m=1}^{M} \tilde{g}_n^2(x_{m_s} + x_m) - 1)^2 \tag{3.25}$$

That completes our specification of the losses. We minimise the full loss with respect to the parameters $(\boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d, \omega_d\}_{d=1}^{D})$ using a gradient-based algorithm, ADAM (Kingma and Ba, 2014). We initialise these parameters by sampling from independent zero-mean gaussians, with variances as in table **??**.

The final detail of our approach is the setting of λ_p and λ_b, which control the relative weight of the constraints vs. the functional objective. We don't want the optimiser to strike a tradeoff between the objective

and constraints, since the constraints must actually be satisfied. But we also don't want to make λ_p so large that the objective is badly minimised in the pursuit of only constraint satisfaction. To balance these demands we use GECO (Rezende and Viola, 2018), which specifies a set of stepwise updates for λ_p and λ_b. These ensure that if the constraint is not satisfied their coefficients increase, else they decrease allowing the optimiser to focus on the loss. The dynamics that implement this are as follows, for a timestep, t.

GECO defines a log-space measure of how well the constraint is being satisfied:

$$L_t = \log(\mathcal{L}_t) - k \tag{3.26}$$

(or small variations on this), where k is a log-threshold that sets the target log-size of the constraint loss in question, and $\mathcal{L}_t$ is the value of the loss at timestep t. L_t is then smoothed:

$$\hat{L}_t = \alpha \hat{L}_t + (1-\alpha) L_t \tag{3.27}$$

And this smoothed measure of how well the constraint is being satisfied controls the behaviour of the coefficient:

$$\lambda_{t+1} = \lambda_t e^{\gamma \hat{L}_t} \tag{3.28}$$

This specifies the full optimisation, parameters are listed in table **??**.

3.B.2 Numerical Optimisation for Finite Spaces

Optimising the representation of finite variables has one added difficulty, and one simplification. Again, we have a parameterised form, e.g. for an angle θ:

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \cos(n_d \theta) + \boldsymbol{b}_d \sin(n_d \theta) \qquad \boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^{D} \in \mathbb{R}^N, n_d \in \mathbb{Z} \tag{3.29}$$

The key difficulty is that each frequency, n_d, has to be an integer, so we cannot optimise it by gradient descent. Similar problems arise for positions on a torus, or sphere. We shall spend the rest of this section outlining how we circumvent this problem. However, we do also have one major simplification. Because the variable is finite, we can easily ensure the representation is non-negative and bounded across the whole space, avoiding the need for an additional normalisation constraint. Further, for the uniform occupancy distributions we consider in finite spaces, we can analytically calculate the neuron norm, and scale the parameters appropriately to ensure it is always 1:

$$\|g_n(\theta)\|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} g_n^2(\theta) d\theta = \|\boldsymbol{a}_0\|^2 + \frac{1}{2} \sum_{d=1}^{D} \|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2 \tag{3.30}$$

$$\tilde{g}_n(\theta) = \frac{g_n(\theta)}{\|g_n(\theta)\|} \tag{3.31}$$

As such, we simply sample a set of angles $\{\theta_m\}_{m=1}^{M}$, and their normalised representations $\{\tilde{\boldsymbol{g}}(\theta_m)\}_{m=1}^{M}$, and compute the appropriate functional and non-negativity loss, as in equations 3.23 & 3.24.

The final thing we must clear up is how to learn which frequencies, n_d, to include in the code. To do this, we make a code that contains many many frequencies, up to some cutoff $D_{\max}$, where $D_{\max}$ is bigger than N:

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D_{\max}} \boldsymbol{a}_d \cos(d\theta) + \boldsymbol{b}_d \sin(d\theta) \qquad \boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^{D_{\max}} \in \mathbb{R}^N \tag{3.32}$$

We then add a term to the loss that forces the code to choose only D of these $D_{\max}$ frequencies, setting all other coefficient vectors to zero, and hence making the code actionable again but allowing the optimiser to do so in a way that minimises the functional loss.

The term that we add to achieve this is inspired by the *representation* theory we used to write these constraints, Section 3.A.2. We create first a $2D_{\max}+1$-dimesional vector, $\mathbf{I}(\theta)$, that contains all the *irreps* i.e. each of the $D_{\max}$ sines and cosines and a constant. We then create a $(2D_{\max}+1) \times (2D_{\max}+1)$-dimensional transition matrix, $\boldsymbol{G}_I(\Delta\theta)$ that is a *representation* of the rotation group in this space: $\boldsymbol{G}_I(\Delta\theta)\mathbf{I}(\theta) = \mathbf{I}(\theta + \Delta\theta)$. $\boldsymbol{G}_I$ can be simply made by stacking 2-by-2 rotation matrices. Then we create the neural code by projecting the frequency basis through a rectangular weight matrix, $\boldsymbol{W}$: $\boldsymbol{g}(\theta) = \boldsymbol{W}\mathbf{I}(\theta)$. Finally, we create the best possible *representation* of the group in the neural space:

$$\boldsymbol{G}(\Delta\theta) = \boldsymbol{W}\boldsymbol{G}_I(\Delta\theta)\boldsymbol{W}^* \tag{3.33}$$

Where $\boldsymbol{W}*$ denotes the Moore-Penrose pseudoinverse. We then learn $\boldsymbol{W}$, which is equivalent to learning $\boldsymbol{a}_0$ and $\{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^{D_{\max}}$.

Representation theory tells us this will not do a perfect job at rotating the neural representation, unless the optimiser chooses $\boldsymbol{W}$ to cut out all but D of the frequencies. As such, we sample a set of shifts $\{\theta_{m_s}\}_{m_s=1}^{M}$, and measure the following discrepancy:

$$\mathcal{L}_{\text{Transformation}} = \frac{1}{M_S M} \sum_{m_S=1}^{M_S} \sum_{m=1}^{M} \|\boldsymbol{G}(\theta_{m_s})\boldsymbol{g}(\theta_m) - \boldsymbol{g}(\theta_m + \theta_{m_s})\|^2 \tag{3.34}$$

Minimising it as we minimised the other constraint terms will force the code to choose D frequencies and hence make the code actionable.

Since calculating the pseudoinverse is expensive, we replace $\boldsymbol{W}^*$ with a matrix $\boldsymbol{B}$ that we also learn by gradient descent. $\boldsymbol{B}$ quickly learns to approximate the pseudoinverse, speeding our computations.

That wraps up our numerical description. We are again left with three loss terms, two of which enforce constraints. This can be applied to any group, even if you can't do gradient descent through their *irreps*, at the cost of limiting the optimisation to a subset of *irreps* below a certain frequency, $D_{\max}$.

$$\mathcal{L} = \mathcal{L}_{\text{functional}} + \lambda_{\text{p}}\mathcal{L}_{\text{non-negativity}} + \lambda_{\text{T}}\mathcal{L}_{\text{Transformation}} \tag{3.35}$$

3.B.3 Parameters Values

We list the parameter values used to generate the grid cells in Figure 3.2B, and show the full population of neurons in figure 3.6.

Parameter	Meaning	Value
σ	neural lengthscale	0.2
l	χ lengthscale	0.5
T	number of gradient steps	150000
N	number of neurons	64
M	number of sampled points every n_{resample} steps	150
M_S	number of room shifts sampled every n_{resample} steps	15
S	standard deviation of normal for shift sampling	3
n_{resample}	number of steps per resample of points	5
λ_{p0}	initial positivity weighting coefficient	0.1
k_p	log positivity target	-9
α_p	positivity target smoothing	0.9
γ_p	positivity coefficient dynamics coefficient	0.0001
λ_{n0}	same set of GECO parameters for norm constraint	0.005
k_n	ditto	4
α_n	ditto	0.9
γ_n	ditto	0.0001
ϵ_w,	coefficient gradient step size	0.1
ϵ_{om}	frequency gradient step size	0.1
β_1	exponential moving average of first gradient moment	0.9
β_2	exponential moving average of second moment	0.9
η	ADAM non-exploding term	1×10^{-8}

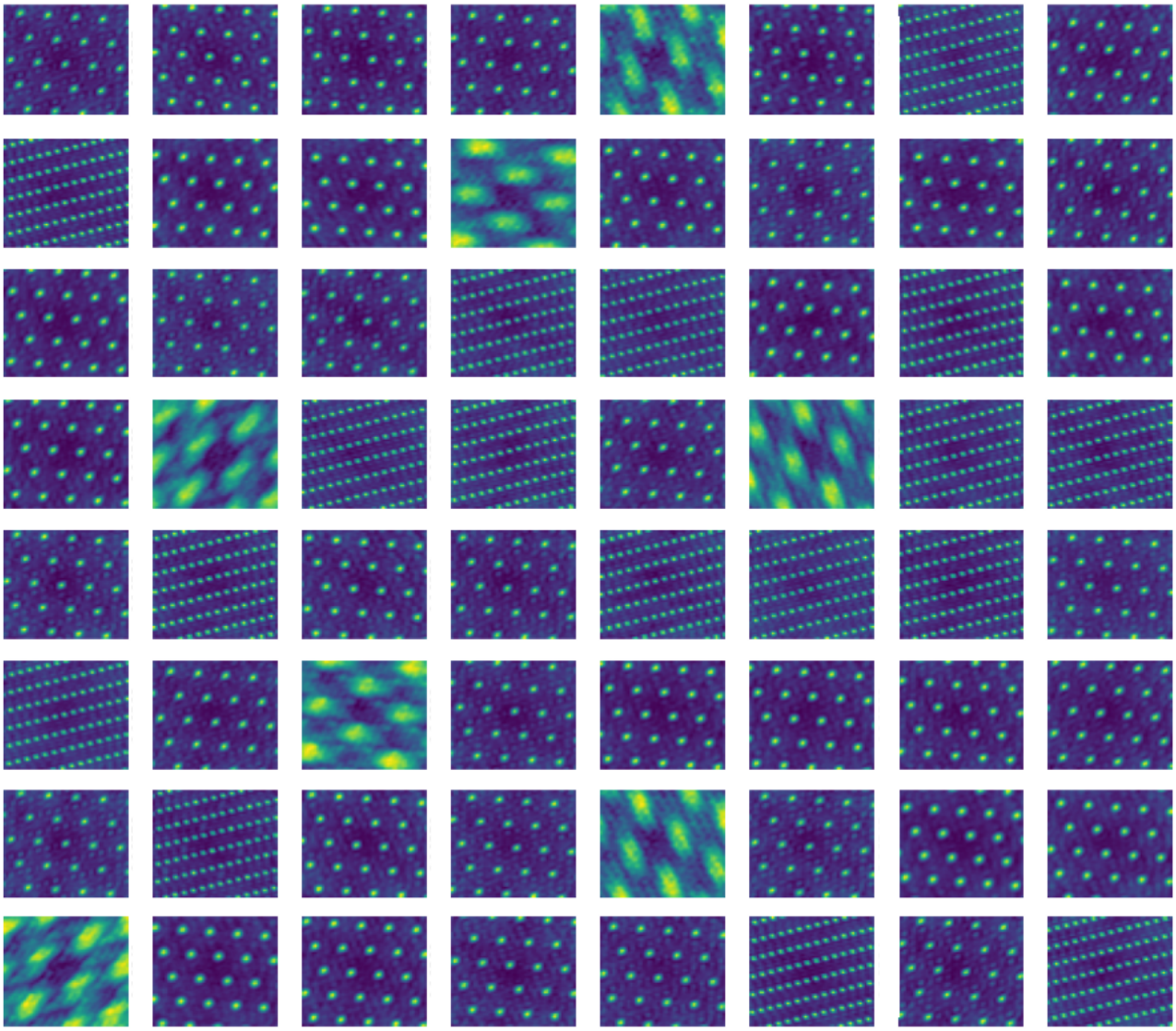


Figure 3.6: All the neurons for the population in figure 3.2: all neurons fall into one of three modules.

3.B.4 Robustness to Parameter Values

In this section we explore the parameter-dependence of our numerical solutions. Most of the parameters in table **??** control the behaviour of the optimisation scheme (for example controlling the behaviour of ADAM (Kingma and Ba, 2014) or GECO (Rezende and Viola, 2018)). These were tuned so that the optimiser found the best solutions possible; any parameter dependence in these does not seem deeply worrying, reflecting the optimisers used rather than the core problem we're solving.

We therefore focus our explorations on the parameters that define key parts of the loss function: the neural lengthscale, σ, the two spatial lengthscales, L and l, and the number of neurons, N. We choose the units of the input space such that $L = 1$, leaving us with three parameters to explore. (The neural space units are not so arbitrary, due to the firing rate constraint)

Our theory actually makes predictions for how these parameters should change the representation. As discussed in appendix 3.E.3, if the ratio of N and σ is small there should be one module of neurons, and increasing it should increase the number of modules in the optimal solutions. l enforces the push to hexagons, so, assuming there's only one module for simplicity, if l is sufficiently large the optimal solution should be hexagonal grid cells. If l is very small then it will have no effect, so all sufficiently high frequency grids should be equal and their shape should matter less.

We verify these claims in a series of numerical experiments, and we show that there are reasonably large parameter regimes in which our suggested qualitative solutions emerge (one module for high σ, as in figure 3.4, multiple modules for low σ, as in figure 3.2). In these experiments most other parameters are kept at the values in table **??**, barring the number of steps which was varied with the number of neurons, and λ_{p0} and λ_{n0} which should be scaled with the approximate size of the loss, that varies as σ and l vary.

Effect of Varying l

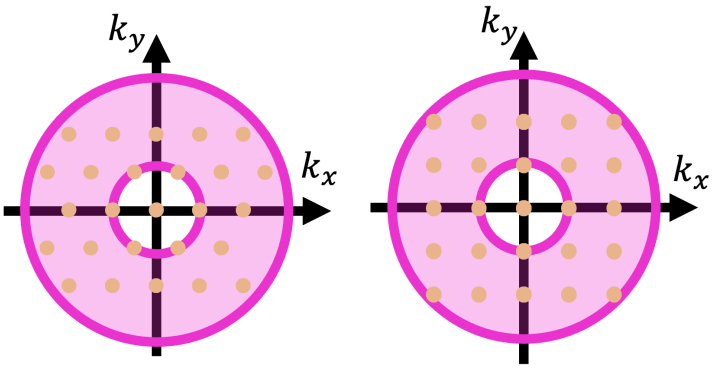


Figure 3.7: For small l, relative to the number of frequencies, many different lattices pack frequencies into the Goldilocks annulus equally well, e.g. these square and hexagonal lattice.

The top 6 panels of figure 3.8 show that for a fixed number of neurons one module of hexagonal grids is optimal for a range of l. We start at $l = 1$, as that corresponds to the rough lengthscale of the environment. Eventually for small l the optimiser chooses other, non-hexagonal grids. While hexagons are optimal for a reasonable range of l, we might wonder why our arguments do not hold for even smaller l?

We believe this is due to the small number of neurons we are using. We argued hexagons were optimal because they packed the most frequencies into a Goldilocks annulus in frequency space, figure 3.4. As l decreases the outer ring of this annulus moves further away, providing a larger Goldilocks region. When the number of neurons in a module, i.e. the number of frequencies, is small this permits many different lattices to pack frequencies within the Goldilocks annulus equally well, figure 3.7. So, for a given number of neurons, there is a l-threshold, beyond which there is no longer a push towards hexagons, and this threshold can be decreased by increasing the number of neurons.

We verify this in the last panel of figure 3.8, where we show that at $l = 0.1$, where 64 neuron modules were not hexagonal, 100 neuron modules were. Therefore, we expect for modules containing many neurons (like those in the brain) hexagons are optimal for a much larger range of l.

Effect of Varying σ

As discussed in Appendix 3.E.3, varying σ varies the push towards non-harmonicity, and hence how many modules we would expect in the optimal solution. We confirm this in Figure 3.9. For large σ (up to $\sigma = 1$) we get one hexagonal module, decreasing it leads to solutions with more modules.

The more modules the less hexagonal they are. We suspect this is again partly a finite neuron effect, as in Appendix 3.B.4. Future work could usefully explore whether more constraints are needed to robustly generate many hexagonal modules, or whether more neurons is enough as it was in figure 3.8.

Varying Number of Neurons

Figures 3.10 and 3.11 show that as we vary the number of neurons (with fixed l), we can find values of σ for which the population produces either one module of hexagonal grids (Figure 3.10) or multiple modules (Figure 3.11). Hence, our qualitative solution types are found across a range of population sizes.

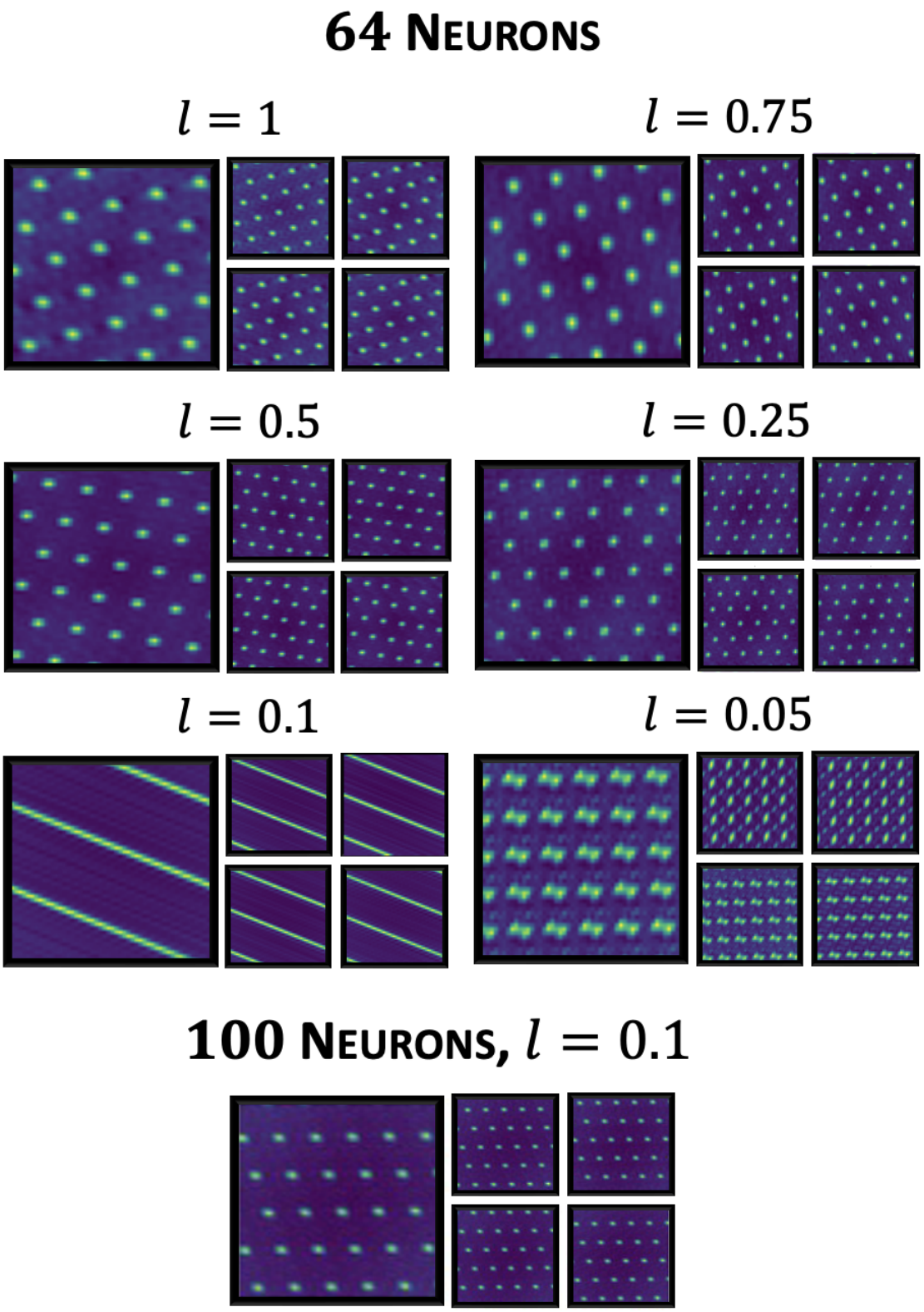


Figure 3.8: For 64 neurons with $\sigma = 0.4$ one module of grids is consistently optimal, all representations contained one module with occasionally one stray neuron with a garbled response. At high l the modules are all hexagonal, for low l other grids start appearing, like band cells or square grids. This is likely due to the small number of neurons used, because when we increase the number of neurons to 100 as in the bottom plot the module remains hexagonal at lower values of l.

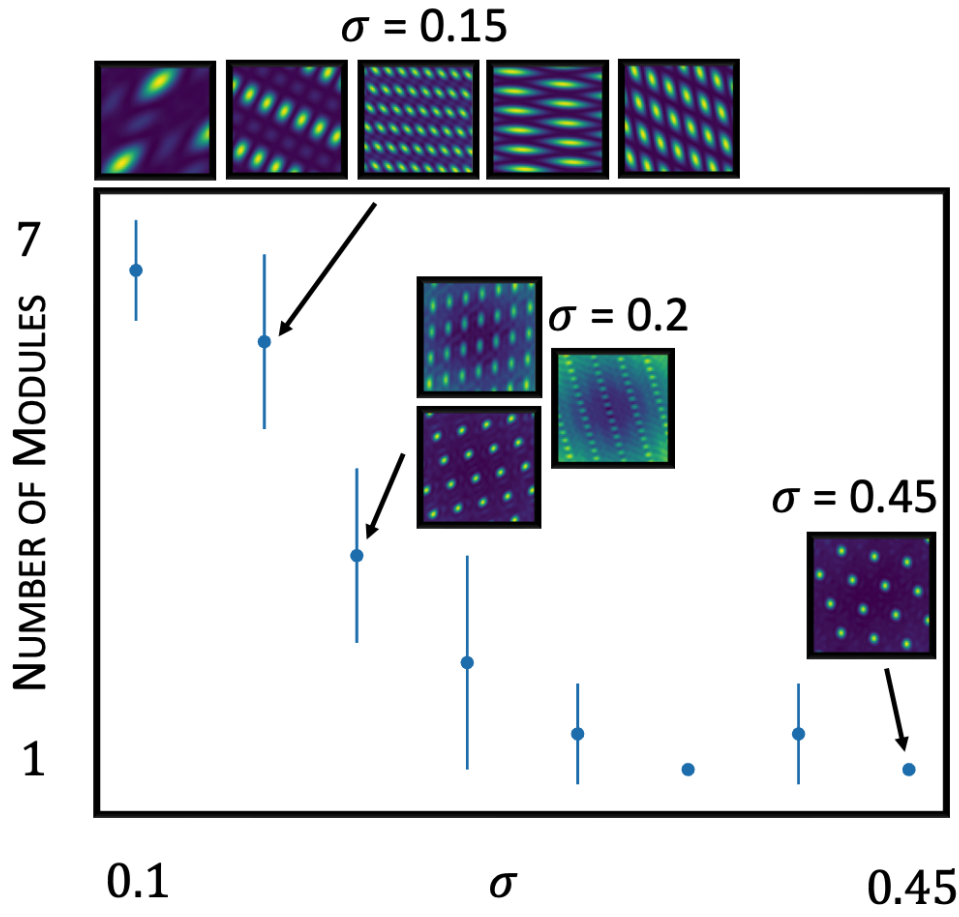


Figure 3.9: For fixed $l = 0.2$ and $N = 64$ we vary σ and count the number of modules. For large σ (including up to $\sigma = 1$) there is consistently one hexagonal module. As σ decreases you get more modules. Error bars are standard deviation over multiple simulations. Inset are the module shapes in one simulation for each of three σ values.

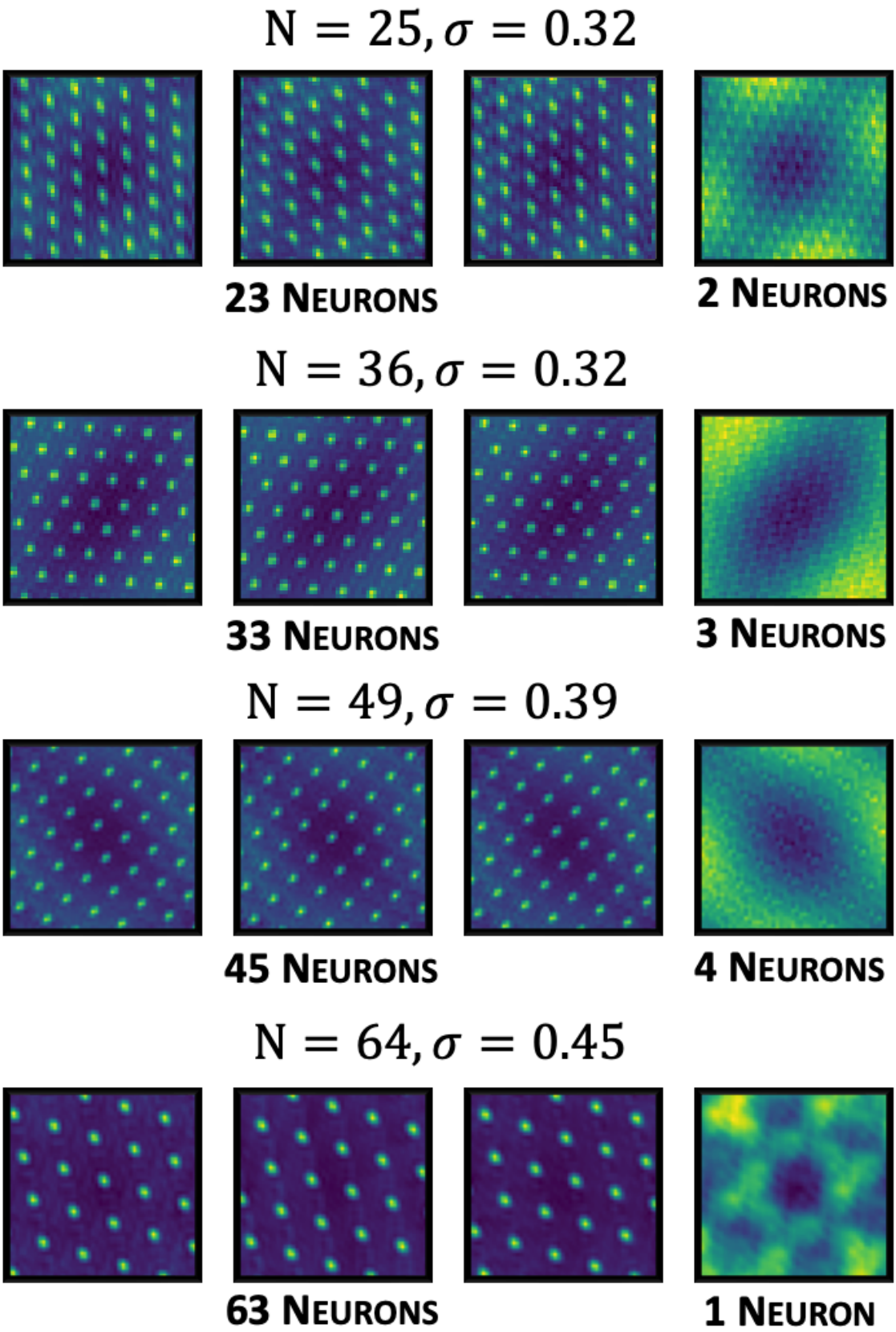


Figure 3.10: Each row of this figure summarises one simulation, and each simulation has a different population size, N. We fix $l = 0.5$ and show a σ value at which almost all neurons form one hexagonal module.

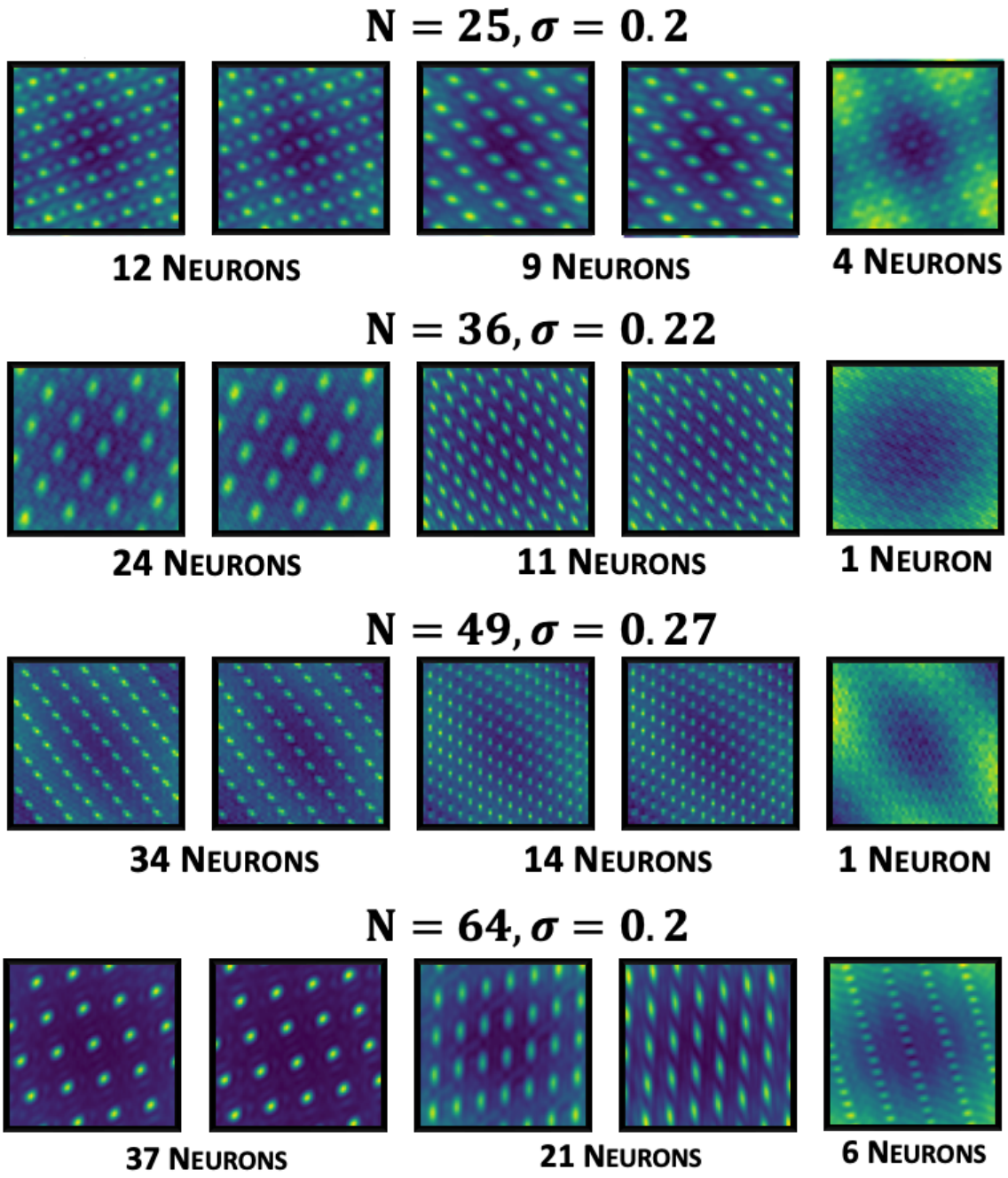


Figure 3.11: Each row of this figure summarises one simulation, and each simulation has a different population size, N. We fix $l = 0.5$ and show a σ value at which the population falls into multiple modules.

3.C Analysis of Simplest Loss that Leads to One Lattice Module

In this section we analytically study the simplified loss suggested in section 3.3.1 and derive equation 3.8. The actionability constraint tells us that:

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(n_d\theta) + \boldsymbol{b}_d \cos(n_d\theta) \qquad n_d \in \mathbb{Z} \tag{3.36}$$

Where D is smaller than half the number of neurons (Appendix 3.A). We ensure $n_d \neq n_{d'}$ if $d \neq d'$ by combining like terms. Our loss is:

$$\mathcal{L}_0 = -\frac{1}{4\pi^2}\iint_{-\pi}^{\pi} \|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2 d\theta d\theta' \tag{3.37}$$

Developing this and using trig identities for the sum of sines and cosines:

$$\mathcal{L}_0 = -\frac{1}{4\pi^2}\iint_{-\pi}^{\pi} \|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2 d\theta d\theta' = -\frac{1}{4\pi^2}\sum_n \iint_{-\pi}^{\pi} (g_n(\theta) - g_n(\theta'))^2 d\theta d\theta' \tag{3.38}$$

$$= -\sum_n \iint_{-\pi}^{\pi} (\sum_d a_{n,d}\cos\left[\frac{n_d(\theta+\theta')}{2}\right]\sin\left[\frac{n_d(\theta-\theta')}{2}\right] - b_{n,d}\sin\left[\frac{n_d(\theta-\theta')}{2}\right]\sin\left[\frac{n_d(\theta-\theta')}{2}\right])^2 \frac{d\theta d\theta'}{\pi^2} \tag{3.39}$$

We now change variables to $\alpha = \frac{\theta-\theta'}{2}$ and $\beta = \frac{\theta+\theta'}{2}$. This introduces a factor of two from the Jacobian $(\frac{1}{2})$, which we cancel by doubling the range of the integral while keeping the same limits, despite the change of variable, Figure 3.12. This gives us:

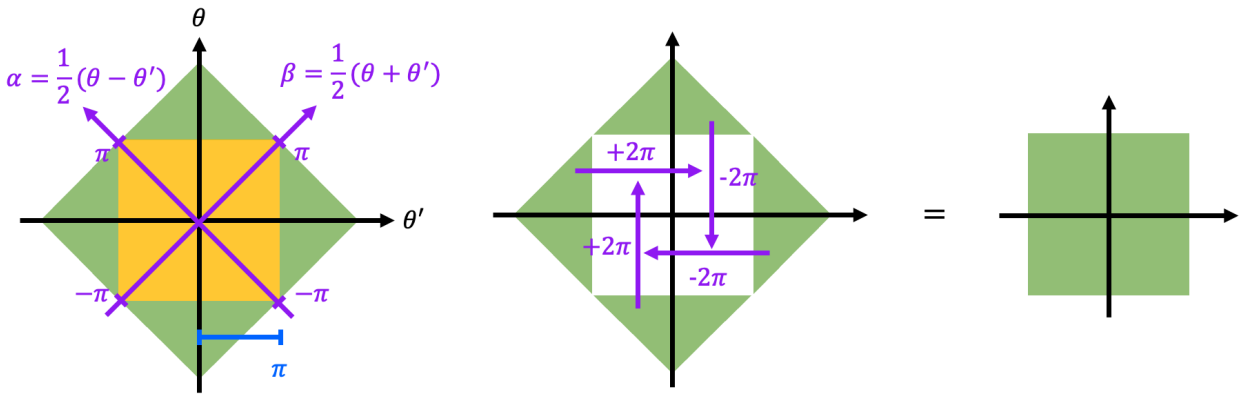


Figure 3.12: The original integral is shown in orange. We rotate to the purple axes and, for convenience, use the same limits (i.e. $-\pi$ to π). This corresponds to adding the green area to the integral but, using the fact that these functions are unchanged by shifts of 2π in either θ or θ' we can see that the green region is equal to the original orange region. Therefore, performing the full purple integral just gets us twice our desired integral.

$$= -\frac{1}{\pi^2}\sum_n \iint_{-\pi}^{\pi} (\sum_d a_{n,d}\cos n_d\beta \sin n_d\alpha - b_{n,d}\sin n_d\beta \sin n_d\alpha)^2 d\alpha d\beta \tag{3.40}$$

$$\begin{aligned} = -\frac{1}{\pi^2}\sum_{n,d,d'} \iint_{-\pi}^{\pi} \Big[& a_{n,d}a_{n,d'}\sin(n_d\alpha)\sin(n_{d'}\alpha)\cos(n_d\beta)\cos(n_{d'}\beta) \\ & - a_{n,d}b_{n,d'}\sin(n_d\alpha)\sin(n_{d'}\alpha)\cos(n_d\beta)\sin(n_{d'}\beta) \\ & - b_{n,d}a_{n,d'}\sin(n_d\alpha)\sin(n_{d'}\alpha)\sin(n_d\beta)\cos(n_{d'}\beta) \\ & + b_{n,d}b_{n,d'}\sin(n_d\alpha)\sin(n_{d'}\alpha)\sin(n_d\beta)\sin(n_{d'}\beta)\Big] d\alpha d\beta \end{aligned} \tag{3.41}$$

Performing these integrals is easy using the fourier orthogonality relations; the two cross terms with mixed sin and cos of β are instantly zero, the other two terms are non-zero only if $n_d = n_{d'}$, i.e. $d = d'$. When $d = d'$ these integrals evaluate to π^2, giving:

$$\mathcal{L}_0 = -\sum_{n,d}(a_{n,d}^2 + b_{n,d'}^2) \tag{3.42}$$

Exactly as quoted in equation 3.7. We can do the same development for the firing rate constraint:

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} g_i^2(\theta)d\theta = a_{0,i}^2 + \frac{1}{2}\sum_{d=1}^{D} a_{d,i}^2 + b_{d,i}^2 = 1, \qquad \forall i = 1, ..., N \tag{3.43}$$

Again using the orthogonality relation. These are the constraints that must be enforced. For illustration it is useful to see one slice of the constraints made by summing over the neuron index:

$$N = \|\boldsymbol{a}_0\|^2 + \frac{1}{2}\sum_{d=1}^{D} \|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2 \tag{3.44}$$

This is the constraint shown in equation 3.8, and is strongly intuition guiding. But remember it is really a stand-in for the N constraints, one per neuron, that must each be satisfied!

This analysis can be generalised to derive exactly the same result in 2D by integrating over pairs of points on the torus. It yields no extra insight, and is no different, so we just quote the result:

$$\mathcal{L}_0 = -\frac{1}{16\pi^4}\int\iiint_{-\pi}^{\pi} \|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}')\|^2 d\boldsymbol{x}d\boldsymbol{x}' = -\sum_d \|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2 \tag{3.45}$$

3.D Analysis of Partially Simplified Loss that Leads to a Module of Hexagonal Grids

In this section we will study an intermediate loss: it will compute the euclidean distance, like in Appendix 3.C, but we'll include χ and p. We'll show that they induce a frequency bias, χ for low frequencies, and p for high, which together lead to hexagonal lattices. We'll study χ first for representations of a circular variable, then we'll study p for representations of a line, then we'll combine them. At each stage generalising to 2-dimensions is a conceptually trivial but algebraically nasty operation with no utility, so we do not detail here. In Appendix 3.I we will return to the differences between 1D and 2D, and show instances where they become important.

3.D.1 Low Frequency Bias: Place Cells on the Circle

We begin with the representation of an angle on a circle, introduce χ, and show the analytic form of the low frequency bias it produces. Since we're on a circle χ must depend on distance on the circle, rather than on a line (i.e. not $\chi = 1 - e^{-\frac{\|x-x'\|^2}{2l^2}}$). We define the obvious generalisation of this to periodic spaces, Figure 3.13A, which is normalised to integrate to one under a uniform $p(\theta)$:

$$\chi(\theta, \theta') = \frac{e^k - e^{k\cos(\theta-\theta')}}{e^k - I_0(k)} \tag{3.46}$$

Where I_0 is the zeroth-order modified Bessel function. Hence the loss is:

$$\mathcal{L}_1 = -\frac{1}{4\pi^2}\iint_{-\pi}^{\pi} \|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2 \frac{e^k - e^{k\cos(\theta-\theta')}}{e^k - I_0(k)} d\theta d\theta' \tag{3.47}$$

In this expression k is a parameter that controls the generalisation width, playing the inverse role of l in the main paper. Taking the same steps as in Appendix 3.C we arrive at:

$$= -\frac{1}{\pi^2}\sum_n \iint_{-\pi}^{\pi} (\sum_d a_{n,d}\cos n_d\beta \sin n_d\alpha - b_{n,d}\sin n_d\beta \sin n_d\alpha)^2 \frac{e^k - e^{k\cos(2\alpha)}}{e^k - I_0(k)} d\alpha d\beta \tag{3.48}$$

The β integral again kills the cross terms:

$$= -\frac{1}{\pi}\sum_{n,d}(a_{n,d}^2 + b_{n,d}^2)\int_{-\pi}^{\pi} \sin^2(n_d\alpha)\frac{e^k - e^{k\cos(2\alpha)}}{e^k - I_0(k)} d\alpha \tag{3.49}$$

These integrals can be computed by relating them to modified Bessel functions of the first kind, which can be defined for integer n as:

$$I_n(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{k\cos(\theta)}\cos(n\theta)d\theta \tag{3.50}$$

Hence the loss is:

$$\mathcal{L}_1 = -\sum_{n,d}(a_{n,d}^2 + b_{n,d}^2)\frac{e^k + I_{n_d}(k)}{e^k - I_0(k)} \tag{3.51}$$

This is basically the same result as before, equation 3.42, all the frequencies decouple and decrease the loss in proportion to their magnitude. However, we have added a weighting factor, that decreases with frequency, Figure 3.13B, i.e. a simple low frequency bias!

This analysis agrees with that of Sengupta et al., 2018, who argued that the best representation of angles on a ring for a loss related to the one studied here should be place cells. We disentangle the relationship between our two works in Appendix 4. Regardless, our optimisation problem now asks us to build a module of lattice neurons with as many low frequencies as possible. The optimal solution to this is the lowest frequency lattice, i.e. place cells! (we validate this numerically in Appendix 3.J). Therefore, it is only additional developments of the loss, namely the introduction of an infinite space for which you prioritise the coding of visited regions, and the introduction of the neural lengthscale, σ, that lead us to conclude that modules of grid cells are better than place cells.

3.D.2 High Frequency Bias: Occupancy Distribution

Now we'll examine how a Gaussian $p(x)$ affects the result, using a representation of x, a 1-dimensional position:

$$\mathcal{L}_1 = -\iint_{-\infty}^{\infty} \|\boldsymbol{g}(x) - \boldsymbol{g}(x')\|^2 p(x)p(x')dxdx' = -\frac{1}{2\pi L^2}\iint_{-\infty}^{\infty} \|\boldsymbol{g}(x) - \boldsymbol{g}(x')\|^2 e^{-\frac{x^2+x'^2}{2L^2}}dxdx' \tag{3.52}$$

Developing the argument as before:

$$= -\frac{4}{\pi L^2}\sum_n \iint_{-\infty}^{\infty}(\sum_d a_{n,d}\cos\omega_d\beta\sin\omega_d\alpha - b_{n,d}\sin\omega_d\beta\sin\omega_d\alpha)^2 e^{-\frac{\alpha^2+\beta^2}{L^2}}d\alpha d\beta \tag{3.53}$$

The cross terms again are zero, due to the symmetry of the β integral around 0.

$$= -\frac{4}{\pi L^2}\sum_{n,d,d'} a_{n,d}a_{n,d'}C(\omega_d,\omega_{d'}\|L)S(\omega_d,\omega_{d'}\|L) + b_{n,d}b_{n,d'}S^2(\omega_d,\omega_{d'}\|L) \tag{3.54}$$

Where we've wrapped up the details into the following integrals of trigonometric functions over a gaussian measure:

$$C(a,b\|L) = \int_{-\infty}^{\infty}\cos(ax)\cos(bx)e^{-\frac{x^2}{L^2}}dx = \frac{L\sqrt{\pi}}{2}(e^{-\frac{L^2(a+b)^2}{4}} + e^{-\frac{L^2(a-b)^2}{4}}) \tag{3.55}$$

$$S(a,b\|L) = \int_{-\infty}^{\infty}\sin(ax)\sin(bx)e^{-\frac{x^2}{L^2}}dx = \frac{L\sqrt{\pi}}{2}(e^{-\frac{L^2(a-b)^2}{4}} - e^{-\frac{L^2(a+b)^2}{4}}) \tag{3.56}$$

So we get our final expression for the loss:

$$\mathcal{L}_1 = -\sum_{n,d,d'} a_{n,d}a_{n,d'}\underbrace{(e^{-\frac{L^2(\omega_d-\omega_{d'})^2}{2}} - e^{-\frac{L^2(\omega_d+\omega_{d'})^2}{2}})}_{\text{Sine Weighting}} + b_{n,d}b_{n,d'}\underbrace{(e^{-\frac{L^2(\omega_d-\omega_{d'})^2}{4}} - e^{-\frac{L^2(\omega_d+\omega_{d'})^2}{4}})^2}_{\text{Cosine Weighting}} \tag{3.57}$$

$$= -\sum_{d,d'}\left[(e^{-\frac{L^2(\omega_d-\omega_{d'})^2}{2}} - e^{-\frac{L^2(\omega_d+\omega_{d'})^2}{2}})\sum_n a_{n,d}a_{n,d'} + (e^{-\frac{L^2(\omega_d-\omega_{d'})^2}{4}} - e^{-\frac{L^2(\omega_d+\omega_{d'})^2}{4}})^2\sum_n b_{n,d}b_{n,d'}\right] \tag{3.58}$$

The downstream consequence of this loss is simple if a module encodes all areas of space evenly. Encoding evenly implies the phases of the neurons in the module are evenly distributed. The phase of a particular neuron is encoded in the coefficients of the sine and cosine of the base frequency, a_{n1} and b_{n1}. Even distribution implies that a_{n1} oscillates from A_d, where $A_d = \max_n a_{n1}$, to $-A_d$ and back again as you sweep the neuron index. As you perform this sweep the subsidiary coefficients, such as a_{n2}, perform the same oscillation, but multiple times. a_{n2} does it twice, since, by the arguments that led us to grids, the peaks of the two waves must align (Figure 3.3B), and that implies the peak of the wave at twice the frequency moves at the same speed as you sweep the neuron number. This implies its phase oscillates twice as fast, and hence that $\sum_n a_{n1}a_{n2} = 0$. Arguments like this imply the frequencies actually decouple for a uniform encoding of space, making the loss,

$$\mathcal{L}_1 = -\sum_{d,n}\left[(1 - e^{-2L^2\omega_d^2})a_{n,d}^2 + (1 - e^{-L^2\omega_d^2})^2 b_{n,d}^2\right] \tag{3.59}$$

This is exactly a high frequency bias! (Fig 3.13C Left)

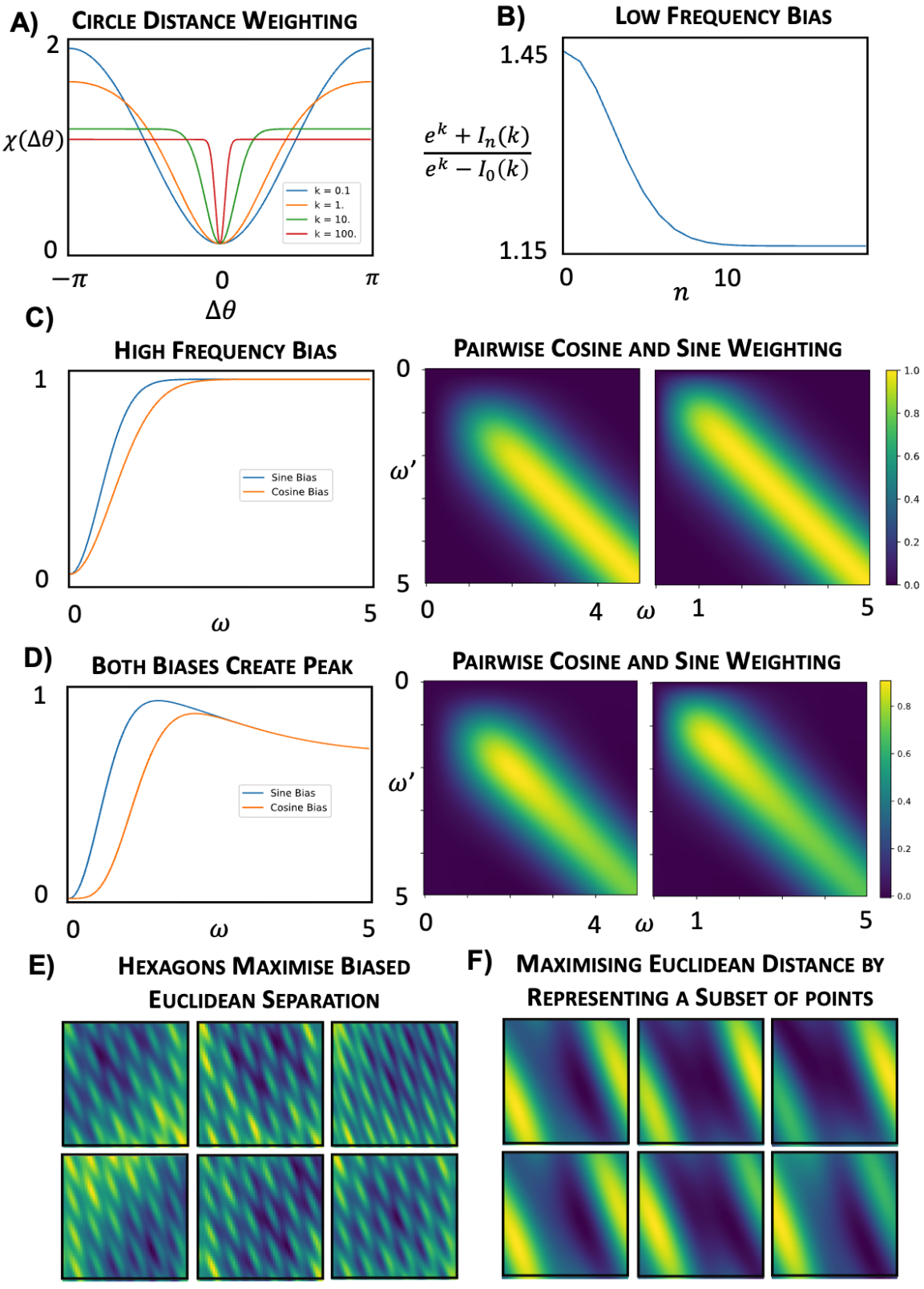


Figure 3.13: **A** The circular version of $\chi(\Delta\theta)$, equation 3.46, weights the importance of separating different pairs on the circle. downweighting nearby points, closer than $\frac{1}{k}$. **B** The Low Frequency Bias of equation 3.51 **C** High Frequency Bias of equation 3.57 and 3.59. Left: bias for isolated frequency (e.g. all frequencies when space is encoded uniformly). Right: cosine and sine bias for a pair of frequencies ω and ω'. Diagonals of right = left. Right plots show both a high frequency bias and a smoothing over similar frequencies, where similar means within $\frac{1}{L}$. (Here $L = 1$) **D** High and Low Frequency Biases. Same as **C** for equation. 3.61. Notice peak around the inverse room size, $\frac{1}{L} = 1$. **E** Optimising a neural code on equation 3.60 with constraints produces one module of hexagons, as predicted. But, as shown in **F**, each neuron activates in the same way, hence this is a bad representation.

However, uniform encoding of the space is actually not optimal for this loss, as can be seen either from the form of equation 3.57, and verified numerically using the loss in the next section, Figure 3.13E-F. The weighting of the sine factor and the cosine factor differ, and in fact push toward sine, since it plateaus at lower frequencies. Combine that with a low frequency push, and you get an incentive to use only sines, which is allowed if you can encode only some areas of space, as the euclidean loss permits.

A final interesting feature of this loss is the way the coupling between frequencies gracefully extends the decoupling between frequencies to the space of continuous (rather than integer) frequencies. In our most simplified loss, equation 3.42, each frequency contributed according to the square of its amplitude, $\|\boldsymbol{a}_d\|^2 + \|\boldsymbol{b}_d\|^2$. How should this be extended to continuous frequencies? Certainly, if two frequencies were very far from one another we should retrieve the previous result, and their contributions would be expected to decouple, $\|\boldsymbol{a}_1\|^2+\|\boldsymbol{b}_1\|^2+\|\boldsymbol{a}_2\|^2+\|\boldsymbol{b}_2\|^2$. But if $\boldsymbol{k}_2 = \boldsymbol{k}_1+\delta\boldsymbol{k}$, for a small $\delta\boldsymbol{k}$ then the two frequencies are, for all intents and purposes, the same, so they should contribute an amplitude $\|\boldsymbol{a}_1+\boldsymbol{a}_2\|^2+\|\boldsymbol{b}_1+\boldsymbol{b}_2\|^2 \neq \|\boldsymbol{a}_1\|^2+\|\boldsymbol{b}_1\|^2+\|\boldsymbol{a}_2\|^2+\|\boldsymbol{b}_2\|^2$. Equation 3.57, Figure 3.13C right, performs exactly this bit of intuition. It tells us that the key lengthscale in frequency space is $\frac{1}{L}$. Two frequencies separated by more than this distance contribute in a decoupled way, as in equation 3.37. Conversely, at very small distances, much smaller than $\frac{1}{L}$ the contribution is exactly $\|\boldsymbol{a}_1 + \boldsymbol{a}_2\|^2 + \|\boldsymbol{b}_1 + \boldsymbol{b}_2\|^2$, as it should be. But additionally, it tells us the functional form of the behaviour between these two limits: a Gaussian decay.

3.D.3 The Combined Effect = High and Low Frequency Bias

In this section we show that the simple euclidean loss with both χ and p can be analysed, and it contains both the high and low frequency biases discussed previously in isolation. This is not very surprising, but we include it for completeness.

$$\mathcal{L}_1 = -\iint_{-\infty}^{\infty} \|\boldsymbol{g}(x) - \boldsymbol{g}(x')\|^2 \underbrace{(1 - e^{-\frac{(x-x')^2}{2l^2}})}_{\chi} \overbrace{\frac{1}{2\pi L^2} e^{-\frac{x^2+x'^2}{2L^2}}}^{p(x)p(x')} dx dx' \tag{3.60}$$

Following a very similar path to before we reach:

$$= -\frac{4}{\pi L^2} \sum_{n,d,d'} a_{nd}a_{nd'}C(\omega_d, \omega_{d'}\|L)\Delta S(\omega_d, \omega_{d'}\|L, \Lambda) + b_{nd}b_{nd'}\Delta S(\omega_d, \omega_{d'}\|L, \Lambda)S(\omega_d, \omega_{d'}\|L) \tag{3.61}$$

$$\Delta S(\omega_d, \omega_{d'}\|L, \Lambda) = S(\omega_d, \omega_{d'}\|L) - S(\omega_d, \omega_{d'}\|\Lambda) \tag{3.62}$$

$$\Lambda = \frac{lL}{\sqrt{l^2 + 2L^2}} \approx \frac{l}{\sqrt{2}} \qquad \text{since} \quad L > l \tag{3.63}$$

This contains the low and high frequency penalisation, at lengthscales $\frac{1}{L}$ and $\frac{1}{l}$, respectively. There are 4 terms, 2 are identical to equation 3.57, and hence perform the same role. The additional two terms are positive, so should be minimised, and they can be minimised by making the frequencies smaller than $\sim \frac{1}{l}$, Figure 3.13D.

We verify this claim numerically by optimising a neural representation of 2D position, $\boldsymbol{g}(\boldsymbol{x})$, to minimise $\mathcal{L}_1$ subject to actionable and biological constraints. It forms a module of hexagonal cells, Figure 3.13E, but the representation is shoddy, because the cells only encode a small area of space well, as can be seen by zooming on the response, Figure 3.13.

3.E Analysis of Full Loss: Multiple Modules of Grids

In this Section we will perturbatively analyse the full loss function, which includes the neural lengthscale, σ. The consequences of including this lengthscale will become most obvious for the simplest example, the representation of an angle on a ring, $\boldsymbol{g}(\theta)$, with a uniform weighting of the importance of separating different points. We'll use it to derive the push towards non-harmonically related modules.

Hence, we study the following loss:

$$\mathcal{L} = \frac{1}{4\pi^2} \iint_{-\pi}^{\pi} e^{-\frac{\|\boldsymbol{g}(\theta)-\boldsymbol{g}(\theta')\|^2}{2\sigma^2}} d\theta d\theta' \tag{3.64}$$

And examine how the effects of this loss differ from the euclidean verison studied in Appendix 3.C.

This loss is difficult to analyse, but insight can be gained by making an approximation. We assume that the distance in neural space between the representations of two inputs depends only on the difference between those two inputs, i.e.:

$$\|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2 = d(\theta - \theta') \tag{3.65}$$

Looking at the constrained form of the representation, we can see this is satisfied if frequencies are orthogonal:

$$\boldsymbol{g}(\theta) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(n_d\theta) + \boldsymbol{b}_d \cos(n_d\theta) \qquad \boldsymbol{a}_0, \{\boldsymbol{a}_d, \boldsymbol{b}_d\}_{d=1}^{D} \in \mathbb{R}^N, n_d \in \mathbb{Z} \tag{3.66}$$

$$\begin{aligned} \boldsymbol{a}_d \cdot \boldsymbol{a}_{d'} &= \delta_{d,d'} A_d^2 \\ \boldsymbol{b}_d \cdot \boldsymbol{b}_{d'} &= \delta_{d,d'} A_d^2 \\ \boldsymbol{a}_d \cdot \boldsymbol{b}_{d'} &= 0 \end{aligned} \tag{3.67}$$

Then:

$$\|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2 = \sum_{d=1}^{D} A_d^2 \sin^2\left(\frac{n_d}{2}(\theta - \theta')\right) \tag{3.68}$$

Now we will combine this convenient form with the following expansion:

$$e^{x\cos(y)} = I_0(x)\Big[1 + 2\sum_{m=1}^{\infty} \frac{I_m(x)}{I_0(x)} \cos(my)\Big] \tag{3.69}$$

Developing equation 3.64 using eqns. 3.69 and 3.68:

$$\mathcal{L} = \frac{1}{4\pi^2} \iint_{-\pi}^{\pi} e^{-\frac{\sum_d A_d^2 \sin^2(n_d \frac{\theta-\theta'}{2})}{2\sigma^2}} d\theta d\theta' \tag{3.70}$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-\frac{\sum_d A_d^2 \sin^2(n_d\alpha)}{2\sigma^2}} d\alpha \tag{3.71}$$

$$= \frac{1}{2\pi} \prod_d \int_{-\pi}^{\pi} e^{-\frac{A_d^2}{4\sigma^2}(1-\cos(2n_d\alpha))} d\alpha \tag{3.72}$$

$$= \frac{1}{2\pi} \underbrace{\prod_d \left[e^{-\frac{A_d^2}{4\sigma^2}} I_0(\frac{A_d^2}{4\sigma^2})\right]}_{\text{Term A}} \underbrace{\int_{-\pi}^{\pi} \prod_d \left[1 + 2\sum_{m=1}^{\infty} \frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})} \cos(2n_d m\alpha)\right] d\alpha}_{\text{Term B}} \tag{3.73}$$

We will now look at each of these two terms separately and derive from each an intuitive conclusion. Term A will tell us that frequency amplitudes will tend to be capped at around the legnthscale σ. Term B will tell us how well different frequencies combine, and as such, will exert the push towards non-harmonically related frequency combinations.

3.E.1 Term A Encourages a Diversity of High Amplitude Frequencies in the Code

Fig 3.14A shows a plot of $I_0(\frac{A_d^2}{4\sigma^2})$. As can be seen there are a couple of regimes, if A_d, the amplitude of frequency n_d, is smaller than σ then $I_0(\frac{A_d^2}{4\sigma^2}) \approx 1$. On the other hand, as A_d grows, I_0 also grows. Asymptotically:

$$I_0(\frac{A_d^2}{4\sigma^2}) \sim \frac{e^{\frac{A_d^2}{4\sigma^2}}}{\sqrt{\frac{A_d^2}{4\sigma^2}}} \tag{3.74}$$

Correspondingly Term A also behaves differently in the two regimes, when A_d is smaller than σ the loss decreases exponentially with the amplitude of frequency d:

$$A_d < \sigma \qquad \text{Term A} = e^{-\frac{A_d^2}{4\sigma^2}} I_0(\frac{A_d^2}{4\sigma^2}) \sim e^{-\frac{A_d^2}{4\sigma^2}} \tag{3.75}$$

Alternatively:

$$A_d > \sigma \qquad \text{Term A} = e^{-\frac{A_d^2}{4\sigma^2}} I_0(\frac{A_d^2}{4\sigma^2}) \sim \frac{2\sigma}{A_d} \tag{3.76}$$

So, up to a threshold region defined by the lengthscale σ, there is an exponential improvement in the loss as you increase the amplitude. Beyond that the improvements drop to being polynomial. This makes a lot of sense, once you have used a frequency to separate a subset of points sufficiently well from one another you only get minimal gains for more increase in that frequency's amplitude, Figure 3.14B. Since there are many frequencies, and the norm constraint ensures that few can have an amplitude of order σ, it encourages the optimiser to put more power into the other frequencies. In other words, amplitudes are only usefully increased to a lengthscale σ, encouraging a diversity in high amplitude frequencies. This is moderately interesting, but will be a useful piece of information for the next derivation.

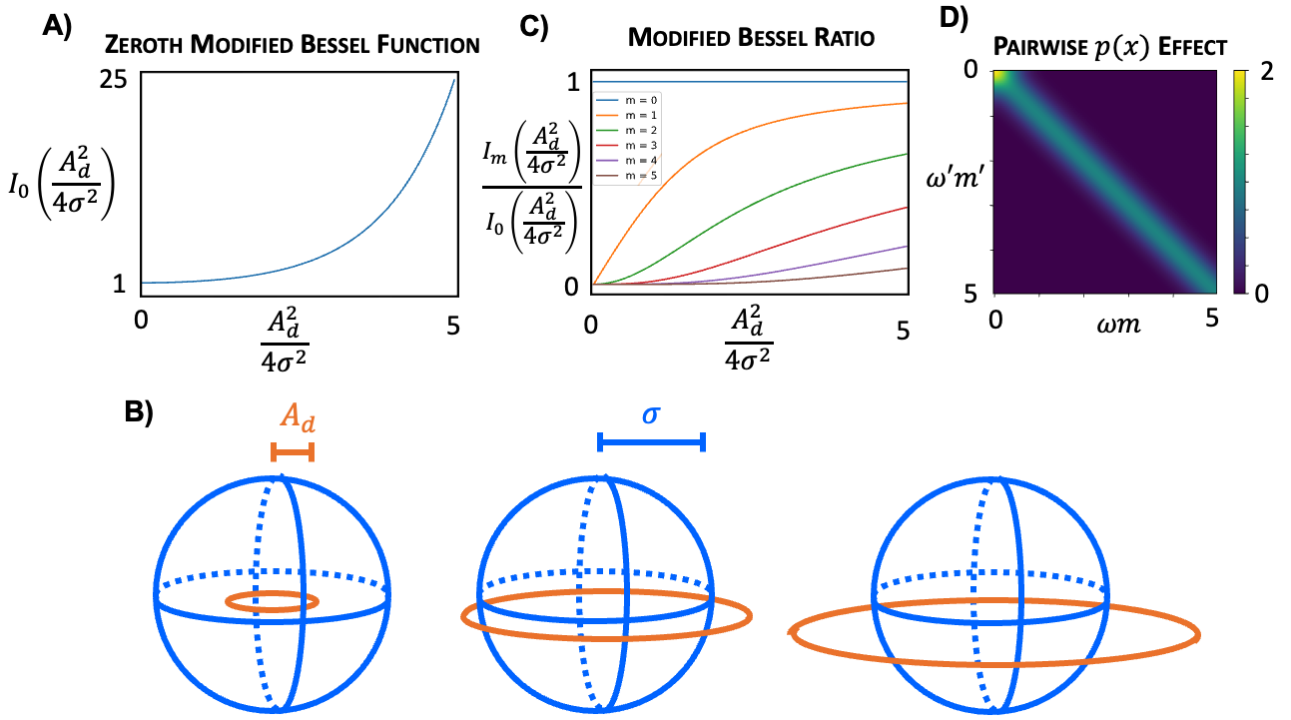


Figure 3.14: **A** Modified Bessel Function. **B** Amplitude Usefully Increases to a limit $\boldsymbol{\sigma}$. When $A_d < \sigma$ then growing the amplitude pushes many points very usefully apart, hence the loss decreases quickly. However, once $A_d \sim \sigma$ many points are already sufficiently separated. The only gains from growing the amplitude more are in the small region that remain neighbouring, hence the loss starts decreasing much more slowly, and precious amplitude is much better invested in other smaller amplitude frequencies. **C** Ratios of Modified Bessel Functions. The terms appearing in the series expansion are, first, smaller than 1, so each additional term is the series, which is multiplied by an additional ratio smaller than 1, decays rapidly. And, second, these ratios decay rapidly with m, so the infinite sums within each term in the series expansion decay rapidly, containing roughly 3 or 4 non-zero terms for the largest amplitude frequencies in the code. **D** Pairwise Continuous Interactions. $p(x)$ causes frequencies to be penalised for both being smaller than $\sim \frac{1}{L}$, and, to a lesser extent, for being harmonically related, up to a lengthscale $\sim \frac{1}{L}$, equation 3.85. This same interaction applies for all pairs of frequencies, and all pairs of harmonics of frequencies, though the contribution is more downweighted the higher the harmonics in the pair.

3.E.2 Term B Encourages the Code's Frequencies to be Non-Harmonically Related

Term B is an integral of the product of D infinite sums, which seems very counter-productive to examine. We are fortunate, however. First, the coefficients in the infinite sum decay very rapidly with m when the ratio of $\frac{A_d}{\sigma}$ is $\mathcal{O}(1)$, Figure 3.14C. The previous section's arguments make it clear that, in any optimal code, A_d will be smaller or of the same order as σ. As such, the inifite sum is rapidly truncated. Additionally, all of the coefficient ratios, $\left(\frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})}\right)$, are smaller than 1, as such we can understand the role of Term B by analysing just the first few terms in the following series expansion,

$$\text{Term B} = \prod_d \left[1 + 2\sum_{m=1}^{\infty} \frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})} \cos(2mn_d\alpha)\right] = \underbrace{1}_{0^{\text{th}}\text{ Order}} \underbrace{+2\sum_{d,m} \frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})} \cos(2mn_d\alpha)}_{1^{\text{st}}\text{ Order}} + \underbrace{4\sum_{d,d',m,m'} \frac{I_m(\frac{A_d^2}{4\sigma^2})I_{m'}(\frac{A_{d'}^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})I_0(\frac{A_{d'}^2}{4\sigma^2})} \cos(2mn_d\alpha)\cos(2m'n_{d'}\alpha)}_{2^{\text{nd}}\text{Order}} + \ldots \tag{3.77}$$

where, due to the rapid decay of $\left(\frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})}\right)$ with m, each term is only non-zero for small m.

The 0^{th} order term is a constant, and the 1^{st} order term is zero when you integrate over α. As such, the leading order effect of this loss on the optimal choice of frequencies arrives via the 2^{nd} order term. The integral

is easy,

$$\int_{-\pi}^{\pi} \cos(2mn_d\alpha)\cos(2mn_{d'}\alpha)d\alpha = \pi\delta(mn_d - m'n_{d'}) \tag{3.78}$$

We want to minimise the loss, and this contribution is positive, so any time the 2nd order term is non-zero it is a penalty. As such, we want as few frequencies to satisfy the condition in the delta function, $mn_d = m'n_{d'}$. Recall from equation 3.77 that these terms are weighted by coefficients that decay rapidly as m grows. As such, the condition that the loss is penalising, $mn_d = m'n_{d'}$, is really saying: penalise all pairs of frequencies in the code, n_d, $n_{d'}$, that can be related by some simple harmonic ratio, $n_d = \frac{m'}{m}n_{d'}$, where m and m' are small integers, roughly less than 5, Figure 3.14C.

This may seem obtuse, but is very interpretable! Simply harmonically related frequencies are exactly those that encode many inputs with the same response! Hence, they should be penalised, Fig 3.4C.

3.E.3 A Harmonic Tussle - Non-Harmonics Leads to Multiple Modules

As described in Section 3.3.3, we now arrive at a dilemma. The arguments of Sections 3.3.1 and 3.C suggest that the optimal representation is comprised of lattice modules, i.e. the neurons must build a code from a lattice of frequencies. A lattice is exactly the arrangement that ensures all the frequencies are related to one another by simple harmonic ratios. Yet, the arguments of the previous section suggest that frequencies with simple harmonic relations are a bad choice - i.e. a frequency lattice would be the worst option! How can this be resolved?

The resolution will depend on the value of two parameters: the first, σ, is the neural lengthscale; the second, N, is the number of neurons, which, through the boundedness constraint, sets the scale of the amplitudes, A_d. We can already see this key ratio, $\frac{A_d}{\sigma}$, appearing in the preceding arguments. Its role is to choose the compromise point in this harmonic tussle, and in doing so it chooses the number of modules in the code.

A simple Taylor expansion illustrates that as σ tends towards ∞ the full loss reverts to the euclidean approximation that was the focus of Sections 3.C and 3.D, in which the effect of the harmonic penalisation drops to zero.

$$e^{-\frac{\|\boldsymbol{g}(\theta)-\boldsymbol{g}(\theta')\|^2}{2\sigma^2}} = 1 - \|\boldsymbol{g}(\theta) - \boldsymbol{g}(\theta')\|^2\frac{1}{2\sigma^2} + \mathcal{O}(\frac{1}{\sigma^4}) \tag{3.79}$$

Hence, as $\sigma \to \infty$ we recover our original optimal solution: all the neurons belong to one module, whose shape is determined by χ and p. This explains why, with few neurons (equivalent to $\sigma \to \infty$), the optimal solution is a single module of perfectly hexagonal grids, fig 3.4B.

But as σ decreases the effect of the harmonic penalisation becomes larger and larger. Eventually, one module is no longer optimal. Yet, all the previous arguments that modularised codes are easily made non-negative, and hence valuable, still apply. The obvious solution is to have two modules. Within each module all the neurons are easily made non-negative, but between modules you can have very non-harmonic relations, trading off both requirements.

As such increasing the ratio $\frac{A_d}{\sigma}$, either by increasing the number of neurons or decreasing σ, creates an optimisation problem for which the high quality representations are modules of grids with more modules as the ratio gets bigger. We observed exactly this effect numerically, and it explains the presence of multiple modules of grids in the full version of our problem for a range of values of σ. Finally, we can extract from this analysis a guiding light: that the frequencies in different modules should be maximally non-harmonically related. This additional principle, beyond the simple low and high frequency biases of Section 3.D, frames our normative arguments for the optimal arrangement of modules, that we explore Section 3.4.2 and Appendix 3.H.

3.E.4 Non-Harmonically Related Continuous Frequencies - A Smoothing Effect

We will address one final concern. When representing infinite variables the frequencies in the code become real numbers. For example, when representing a 1-dimensional continuous input, x, the code, $\boldsymbol{g}(x)$, has the form,

$$\boldsymbol{g}(x) = \boldsymbol{a}_0 + \sum_{d=1}^{D} \boldsymbol{a}_d \sin(\omega_d x) + \boldsymbol{b}_d \cos(\omega_d x) \qquad \omega_d \in \mathbb{R} \tag{3.80}$$

And it seems like the equivalent penalisation as in equation 3.78, $\delta(m\omega_d - m'\omega_{d'})$, is very easy to avoid. If $m\omega_d = m'\omega_{d'}$ and the code is being penalised, just increase ω_d by an infinitely small amount, $\delta\omega$, such that $m\omega_d + m\delta\omega \neq m'\omega_{d'}$, and suddenly this term does not contribute to the loss.

This, however, is not how the loss manifests. Instead frequencies are penalised for being close to one another, where close is defined by the size of the region the animal encodes, set by lengthscale L. This can be seen by

analysing the following loss,

$$\mathcal{L} = \iint_{-\infty}^{\infty} e^{-\frac{\|\boldsymbol{g}(x)-\boldsymbol{g}(x')\|^2}{2\sigma^2}} p(x)p(x')dxdx' \tag{3.81}$$

$$= \frac{1}{2\pi L^2} \iint_{-\infty}^{\infty} e^{-\frac{\|\boldsymbol{g}(x)-\boldsymbol{g}(x')\|^2}{2\sigma^2}} e^{-\frac{x^2+x'^2}{2L^2}} dxdx' \tag{3.82}$$

Making the same assumption as in the previous Section, equation 3.67, and developing the loss identically,

$$\mathcal{L} = \frac{1}{2\sqrt{\pi}L} \underbrace{\prod_d \left[e^{-\frac{A_d^2}{4\sigma^2}} I_0(\frac{A_d^2}{4\sigma^2}) \right]}_{\text{Term A}} \underbrace{\int_{-\pi}^{\pi} \prod_d \left[1 + 2\sum_{m=1}^{\infty} \frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})} \cos(2\omega_d m\alpha) \right] e^{-\frac{\alpha^2}{4L^2}} d\alpha}_{\text{Term B}} \tag{3.83}$$

We again analyse this through the series expansion in equation 3.77. The 0^{th} order term is again constant. The 1^{st} order term is proporational to,

$$\frac{1}{2\sqrt{\pi}L} \int_{-\infty}^{\infty} \cos(2m\omega_d\alpha) e^{-\frac{\alpha^2}{4L^2}} d\alpha = e^{-4L^2m^2\omega_d^2} \tag{3.84}$$

i.e., as in Section 3.D.2, $p(x)$ implements a high frequency bias. In fact, the mathematical form of this penalty, a gaussian, is identical to that in diagonal terms of the previously derived sum, equation 3.57.

The 2^{nd} order term is where we find the answer to this Section's original concern,

$$\frac{1}{2\sqrt{\pi}L} \int_{-\infty}^{\infty} \cos(2m\omega_d\alpha) cos(2m'\omega_{d'}\alpha) e^{-\frac{\alpha^2}{4L^2}} d\alpha = \frac{1}{2\sqrt{\pi}L} C(2m\omega_d, 2m'\omega_{d'} \| 2L) \tag{3.85}$$

$$= \frac{1}{2}(e^{-4L^2(m\omega_d+m'\omega_{d'})^2} + e^{-4L^2(m\omega_d-m'\omega_{d'})^2}) \tag{3.86}$$

Which is similar to the cross terms in equation 3.57, but with the addition of harmonic interactions. This is the continuous equivalent to the penalty in equation 3.78. Frequencies and their low order harmonics are penalised for landing closer than $\sim \frac{1}{L}$ from one another, Fig 3.14D. In conclusion, the logic gracefully transfers to continuous frequencies

3.F Ablation Studies

We have argued that three principles - biological, functional, and actionable - are necessary to produce grid cells. In this section we show that removing any of these three and numerically optimising produces fundamentally different patterns, and we explain these results with the theory we've developed in this work.

We study the representation of a small torus, since for periodic spaces the three terms appear cleanly in the loss, so can be easily excised, Appendix 3.B. Figure 3.15 shows that removing any term stops grid cells emerging.

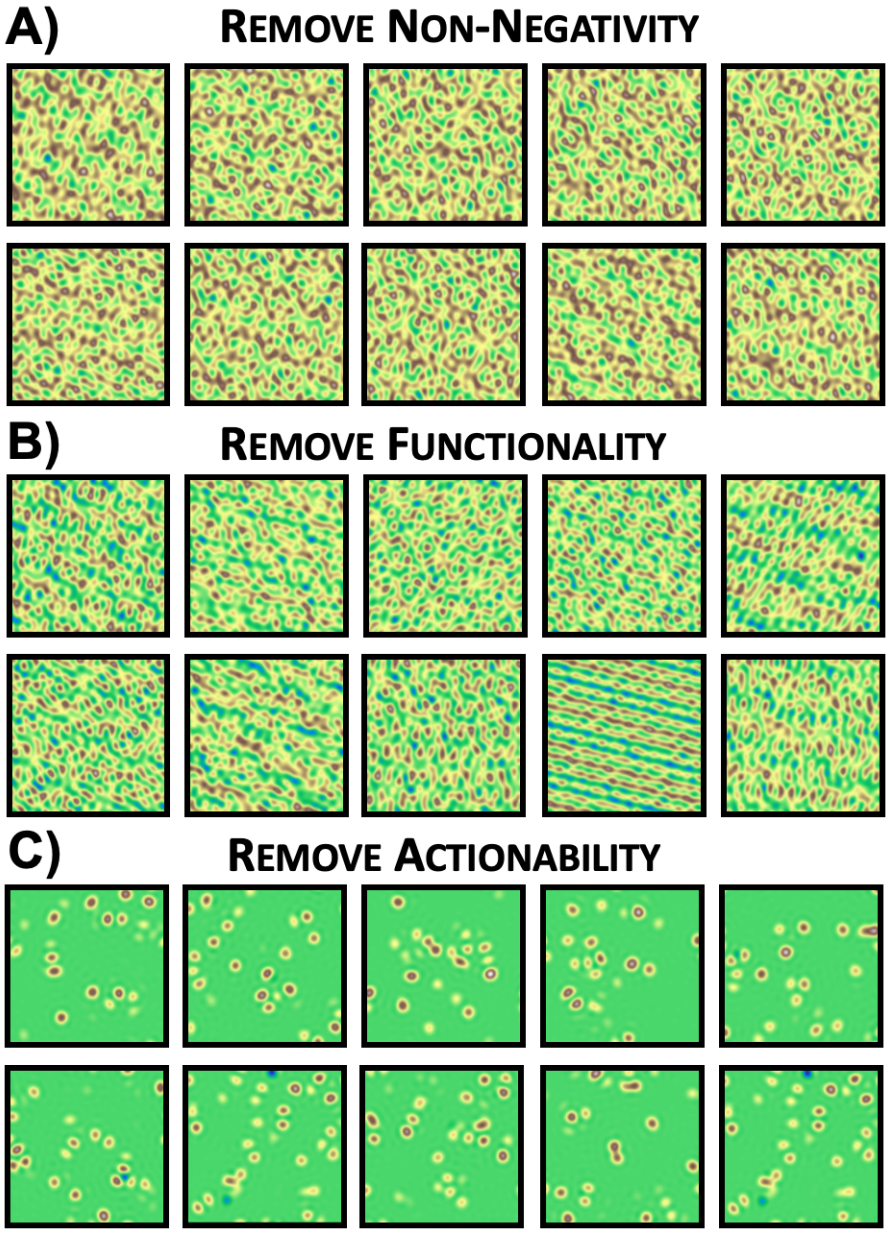


Figure 3.15: **A** Without non-negativity the code is simply built from D non-harmonically related frequencies. **B** Without the functional term an optimal code can be made by increasing the constant vector sufficiently, and containing few frequencies, leaving a random pattern of sines and cosines. **C** Without actionability there is no push towards sparsity in fourier space. Bumps are still easy to positivise, without frequency sparsity there's no reason to arrange them into lattices. Hence, this response looks a bit like grid cells with the peak positions randomised.

No Non-negativity Without non-negativity the code wants to maximise the size of each of the limited number of frequencies in the code, ensuring that they are non-harmonically related. As such, it builds a wavey but fairly random looking code.

No Functional Without a functional term the loss simply wants the code to contain few frequencies and be non-negative, so produces random looking responses with a sparse fourier spectrum and a large constant offset to ensure non-negativity.

No Actionability Without actionability the code wants to be non-negative but separable. Bumps are easily positivised and encode the space well, but there is no reason to arrange them into lattices, and it would actually be detrimental to co-ordinate the placement across neurons so that any hypothetical lattice axes align. As such, it builds a random but bumpy looking pattern. This observation will be important for our discussion of the coding of 3-dimensional space, Appendix 3.J.

3.G Lattice Size:Peak Width Ratio Scales with Number of Neurons

Thanks to the actionability constraint, the number of neurons in a module controls the number of frequencies available to build the module's response pattern. In our scheme the frequencies within one module organise into a lattice, and the more frequencies in the module, the higher the top frequency, and the correspondingly sharper the grid peaks. This link allows us to extract the number of neurons from a single neuron's tuning curve in the module. In this section we add slightly more justification to this link.

We begin with a representation of 1D position, $\boldsymbol{g}(x)$. When there are enough neurons the griddy response can be approximated as,

$$g_n(x) \approx \mathcal{N}(x \| 0, \mu) * DC(\nu), \tag{3.87}$$

where $DC(\nu)$ is a Dirac comb with lengthscale ν, $*$ is convolution, and μ is the grid peak width, Fig 3.16. The fourier transform of this object is simple, Fig 3.16,

$$\tilde{g}_n(\omega) \approx \mathcal{N}\left(\omega \| 0, \frac{1}{\mu}\right) \times DC\left(\frac{1}{\nu}\right) \tag{3.88}$$

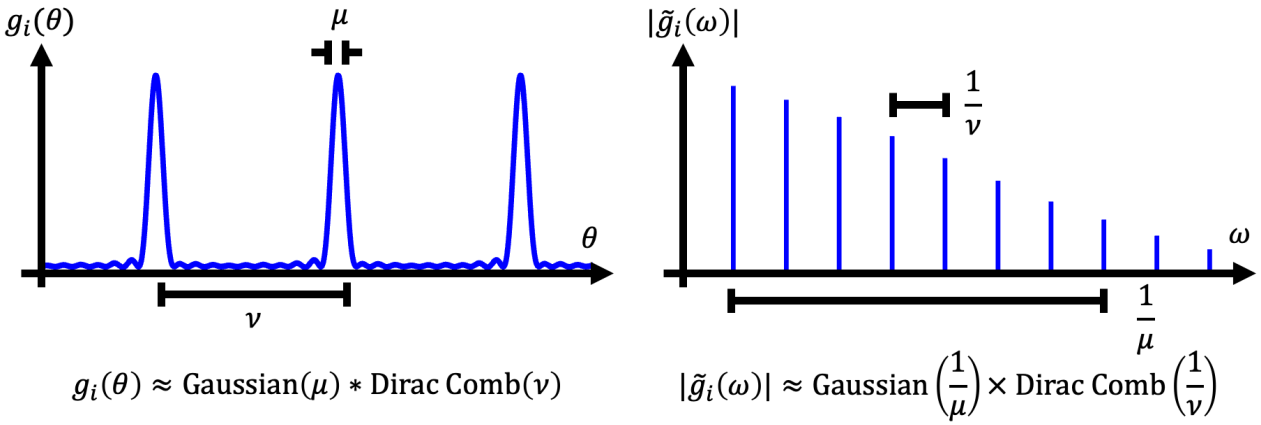


Figure 3.16: Approximate Large Neuron Responses. With many neurons the grid response is approximately a dirac comb convolved with a Gaussian, and hence the frequency response is approximately a Gaussian multiplied by a Dirac Comb. Actionability implies sparsity in fourier space, which we approximate as a finite number of frequencies having power above some threshold τ, i.e. a sufficiently fast decay in frequency space, which corresponds to a sufficiently broad grid peak width.

We can now extract the tradeoff exactly. In this continuous approximation the equivalent of actionability's sparsity in frequency space is that the frequency spectra must decay sufficiently fast i.e. the number of frequencies with power above some small threshold must be less than half the number of neurons. This number is controlled by the ratio of μ and ν. We'll set some arbitrary low scaled threshold, τ, which defines the power a frequency must have, relative to the constant component, to be considered as a member of the code. The actual value of τ is inconsequential, as we just want scaling rules. Then we'll derive the scaling of the frequency at which the amplitude of the gaussian drops below threshold, ω^*:

$$\frac{\|\mathcal{N}(\omega^* \| 0, \frac{1}{\mu})\|^2}{\|\mathcal{N}(0 \| 0, \frac{1}{\mu})\|^2} = e^{-\mu^2 \omega^{*2}} = \tau \tag{3.89}$$

$$\omega^* = \frac{\log(\frac{1}{2\tau})}{\mu} \tag{3.90}$$

Now we can ask how many frequencies there are in the code, by counting the number of peaks of the Dirac comb that occur before this threshold. Counting from 0, because in 1D negative frequencies are just scaled

versions of positive frequencies:

$$D = \frac{\omega^*}{\frac{1}{\nu}} = \frac{\nu}{\mu}\log\left(\frac{1}{2\tau}\right) \tag{3.91}$$

Therefore, in 1D, when there are enough neurons, the number of frequencies, and therefore the number of the neurons scales with the lattice to peak width ratio:

$$N \propto \frac{\nu}{\mu} \tag{3.92}$$

A completely analogous argument can be made in 2D, Again the cutoff frequency is $\omega^* \propto \frac{1}{\mu}$. A different scaling emerges, however, because the number of frequencies in a dirac comb lower than some threshold scales with dimension. The frequency density of the dirac comb is proportional to $\frac{1}{\nu^2}$ peaks per unit area with the specifics depending on the shape of the frequency lattice, so we're asking how many peaks are there in a half circle of radius ω^*. This is simple,

$$D \propto \frac{\pi\omega^{*2}}{2\frac{1}{\nu^2}} = \frac{\nu^2}{\mu^2}\frac{\pi}{2}\log\left(\frac{1}{2\tau}\right) \tag{3.93}$$

Hence we recover the stated scaling,

$$N \propto \left(\frac{\nu}{\mu}\right)^2 \tag{3.94}$$

3.H Optimal Relative Angle Between Modules

Our analysis of the full loss, Appendix 3.E, suggested that optimal representations should contain multiple modules, and each module should be chosen such that its frequencies are optimally non-harmonically related to the frequencies in other modules. This is a useful principle, but the full implications are hard to work through exactly. In this section we use rough approximations and coarse calculations to make estimates of one aspect of the relationships between modules: the optimal relative angle between lattice axes. Our scheme begins by calculating a smoothed frequency lattice density as a function of angle, $\rho(\psi)$. It then aligns two of these smoothed densities, representing two modules, at different angular offsets, and finds the angle which minimises the overlap. This angle is the predicted optimal angle. We then extend this to multiple modules by searching over all doubles or triples of alignment angles for the minimal overlap of three or four densities. We will talk through the details of this method, and arrive at the results that are quoted in section 3.4.2.

If you imagine standing at the origin in frequency space, corresponding to the constant component, and looking to infinity at an angle ψ to the k_x axis, the density function, $\rho(\psi)$, measures, as a function of ψ, the number of smoothed frequencies that you see, fig 3.17A. We assume there are many neurons in this module, and hence many frequencies in the frequency lattice. Then we make the same approximation as in eqns. 3.87 and 3.88, that the frequency response is a hexagonal grid multiplied by a decaying Gaussian. The lengthscale of this Guassian is the first parameter that will effect our results, and it scales linearly with the number of neurons in the module. Then we try and capture the repulsive effects of the lattice. Each lattice point repels frequencies over a nearby region, as calculated in Section 3.E.4. This region is of size $\sim \frac{1}{2L}$, where L is the exploration lengthscale of the animal. We therefore smooth the grid peaks in our density function by convolving the lattice with a Gaussian. We choose our units such that $L = 1$, then the width of this smoothing Gaussian is also order 1. The second important parameter is then the grid lattice lengthscale. The larger the grid cell lengthscale the more overlapping the repulsive effects of the frequencies will be, and the smoother the repulsive density, $\rho(\psi)$. In maths,

$$\rho(\psi\|\mu,\nu) = \int_0^\infty \left[\mathcal{N}\left(\boldsymbol{k}\middle|0,\frac{1}{\mu}\right) \times DC\left(\boldsymbol{k}\middle|\frac{1}{\nu}\right)\right] * \mathcal{N}(\boldsymbol{k}\|0,1)k_r dk_r \tag{3.95}$$

where k_r is the radial part of the frequency. We calculate the obvious numerical approximation to this integral, Fig 3.17B, and in these calculations we ignore the effect of higher harmonics in equation 3.E.4. This isn't as bad as is sounds, since higher harmonics contribute density at the same angle, ψ, part of the reason we chose to calculate this angular density.

The next stage is finding the optimal angle between modules, starting with two identical modules. We do this by overlapping copies of $\rho(\psi\|\mu,\nu)$. For simplicity, we start with two identical lattices and calculate the overlap as a function of angular offset:

$$\tilde{\rho}_2(\Delta\psi\|\mu,\nu) = \int_{-\pi}^{\pi} \rho(\psi\|\mu,\nu)\rho(\psi+\Delta\psi\|\mu,\nu)d\psi \tag{3.96}$$

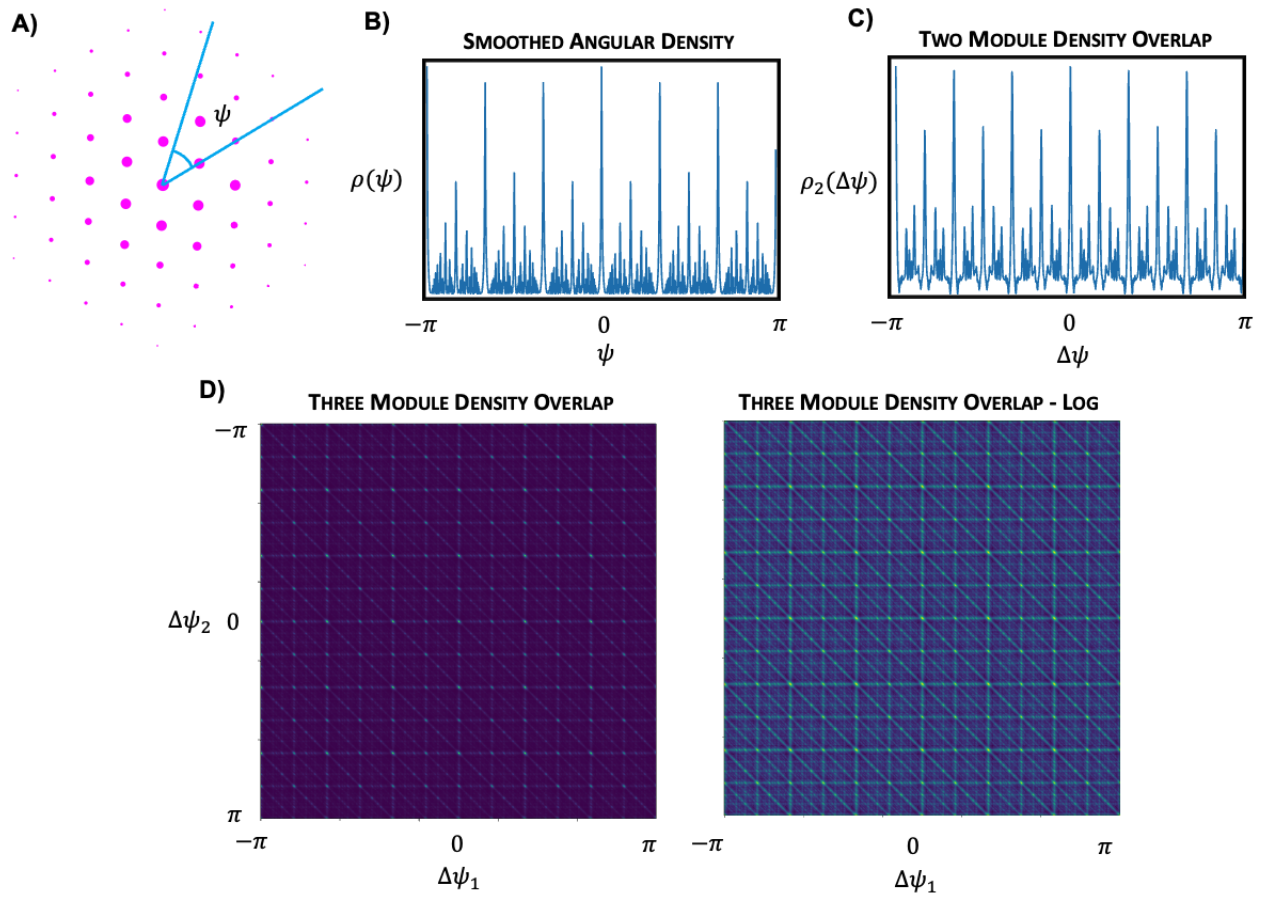


Figure 3.17: **A: Smoothed Angular Density** We stand at the origin and look out at angle ψ. We then count the density of smoothed grid frequency peaks along this line, where the smoothing occurs over lengthscale $\frac{1}{L}$. **B: Smoothed Density Plot. C: Overlap of Density Plots at Offsets $\Delta\psi$** This curve shows a clear minima at 4 degrees from perfect alignment of grid axes. **D: Three Module Overlaps** The overlap (left) and the log of the overlap (right) show similar patterns as a function of the two angular offsets between modules, and the overlap is minimise for stepwise offsets of around 4 degrees.

Then the predicted angular offset is,

$$\Delta\phi^* = \arg\min_{\Delta\phi} \tilde{\rho}_2(\Delta\psi \| \mu, \nu) \tag{3.97}$$

Figure 3.17C shows this is a small angular offset around 4 degrees, though it can vary between 3 and 8 degrees depending on μ and ν. An identical procedure can be applied to multiple modules. For example, for the arrangement of three modules we define,

$$\tilde{\rho}_3(\Delta\psi, \Delta\psi' \| \mu, \nu) = \int_{-\pi}^{\pi} \rho(\psi \| \mu, \nu)\rho(\psi + \Delta\psi \| \mu, \nu)\rho(\psi + \Delta\psi + \Delta\psi' \| \mu, \nu)d\psi \tag{3.98}$$

and find the pair of offsets that minimise the overlap. For identical modules this also predicts small offsets between modules, of around 4 degrees between each module, Figure 3.17D, and similarly three identical modules should be oriented at multiples of 4 degrees from a reference lattice (i.e. offsets of 4, 8, and 12). As shown in Figure 3.5E, these small offsets appear to be present in data, validating our prediction.

There are many natural next steps for this kind of analysis. In future we will explore the effect of different lattice lengthscales and try to predict the optimal module arrangement. Another simple analysis you can do is explore how changing the grid lengthscale, ν, effects the optimal relative angle. The larger the grid lattice lengthscale, the smaller its frequency lattice, and the more smooth the resulting repulsion density is. As a result, the optimal relative angle between large modules is larger than between small modules, a prediction we would like to test in data. It would be interesting to apply more careful numerical comparisons, for example matching the ν and μ to those measured in data, and comparing the real and optimal angular offsets.

3.I Room Geometry Analysis

The functional objective depends on the points the animal wants to encode, which we assume are the points the animal visits. As such, the exploration pattern of the animal determines the optimal arrangement of grids, through the appearance of $p(x)$ in the functional objective. Different constraints to the exploration of the animal, such as different sized environments, will lead to different optimal representations. In this section we analyse the functional objective for a series of room shapes, and show they lead to the intricate frequency biases suggested in Section 3.4.3.

3.I.1 1-Dimensonal Box

To begin and to set the pattern for this section, we will study the representation of 1-dimensional position in a finite box. Ignoring the effects of χ for now, our loss is,

$$\mathcal{L} = \frac{1}{L^2}\iint_{-\frac{L}{2}}^{\frac{L}{2}} e^{-\frac{\|\boldsymbol{g}(x)-\boldsymbol{g}(x')\|^2}{2\sigma^2}}dxdx' \tag{3.99}$$

Making the symmetry assumption as in Section 3.E our loss is,

$$\mathcal{L} = \frac{1}{L^2}\iint_{-\frac{L}{2}}^{\frac{L}{2}} e^{-\frac{\sum_d A_d^2 \sin^2(\frac{\omega_d(x-x')}{2})}{2\sigma^2}}dxdx' \tag{3.100}$$

Changing variables to $\alpha = x - x'$, $\beta = x + x'$, Figure 3.18A,

$$\mathcal{L} = \frac{1}{2L^2}\left[\int_0^L\int_{-L+\alpha}^{L-\alpha} e^{-\frac{\sum_d A_d^2 \sin^2(\frac{\omega_d\alpha}{2})}{2\sigma^2}}d\beta d\alpha + \int_{-L}^0\int_{L+\alpha}^{-L-\alpha} e^{-\frac{\sum_d A_d^2 \sin^2(\frac{\omega_d\alpha}{2})}{2\sigma^2}}d\beta d\alpha\right] \tag{3.101}$$

$$= \frac{1}{L^2}\int_0^L e^{-\frac{\sum_d A_d^2 \sin^2(\frac{\omega_d\alpha}{2})}{2\sigma^2}}(L-\alpha)d\alpha \tag{3.102}$$

Performing the expansion as in Section 3.E we reach:

$$\mathcal{L} = \frac{1}{L^2}\underbrace{\prod_d\left[e^{-\frac{A_d^2}{4\sigma^2}}I_0(\frac{A_d^2}{4\sigma^2})\right]}_{\text{Term A}}\underbrace{\int_{-\pi}^{\pi}\prod_d\left[1+2\sum_{m=1}^{\infty}\frac{I_m(\frac{A_d^2}{4\sigma^2})}{I_0(\frac{A_d^2}{4\sigma^2})}\cos(\omega_d m\alpha)\right](L-\alpha)d\alpha}_{\text{Term B}} \tag{3.103}$$

Term A behaves the same as before, so we again study the series expension of term B, the analogue of equation 3.77. The 0^{th} order term is a constant with respect to the code, but the first order term provides the main frequency biasing effect, as in Section 3.E.4.

$$\frac{1}{L^2}\int_0^L \cos(m\omega_d\alpha)(L-\alpha)d\alpha = \frac{1}{m^2L^2\omega_d^2}(1-\cos(m\omega_d L)) \tag{3.104}$$

Hence the 1-dimensional box coarsely creates a high frequency bias ($> \frac{1}{L}$), but also encourages frequencies that fit whole numbers of wavelengths within the length of the box, Figure 3.18B.

Before we move on we will study the second order term. We won't repeat this for future finite rooms as it shows basically the same effect as in Section 3.E.4: coarsly frequencies repel each other up to a lengthscale $\frac{1}{L}$. However, as for the first order term, there is additional structure, i.e. subsidiary optimal zeros in the loss, that could be of interest:

$$\begin{aligned}\frac{2}{L^2}\int_0^L &\cos(m\omega_d\alpha)\cos(m'\omega_{d'}\alpha)(L-\alpha)d\alpha \\ &= \frac{1-\cos(L(m\omega_d+m'\omega_{d'}))}{(m\omega_d+m'\omega_{d'})^2L^2} + \frac{1-\cos(L(m\omega_d-m'\omega_{d'}))}{(m\omega_d-m'\omega_{d'})^2L^2}\end{aligned} \tag{3.105}$$

This is fundamentally a repulsion between frequencies closer than $\frac{1}{L}$ from one another, fig 3.18C. Additionally, the repulsion is zero when the difference between frequencies is exactly $\frac{2\pi}{L}$ times an integer.

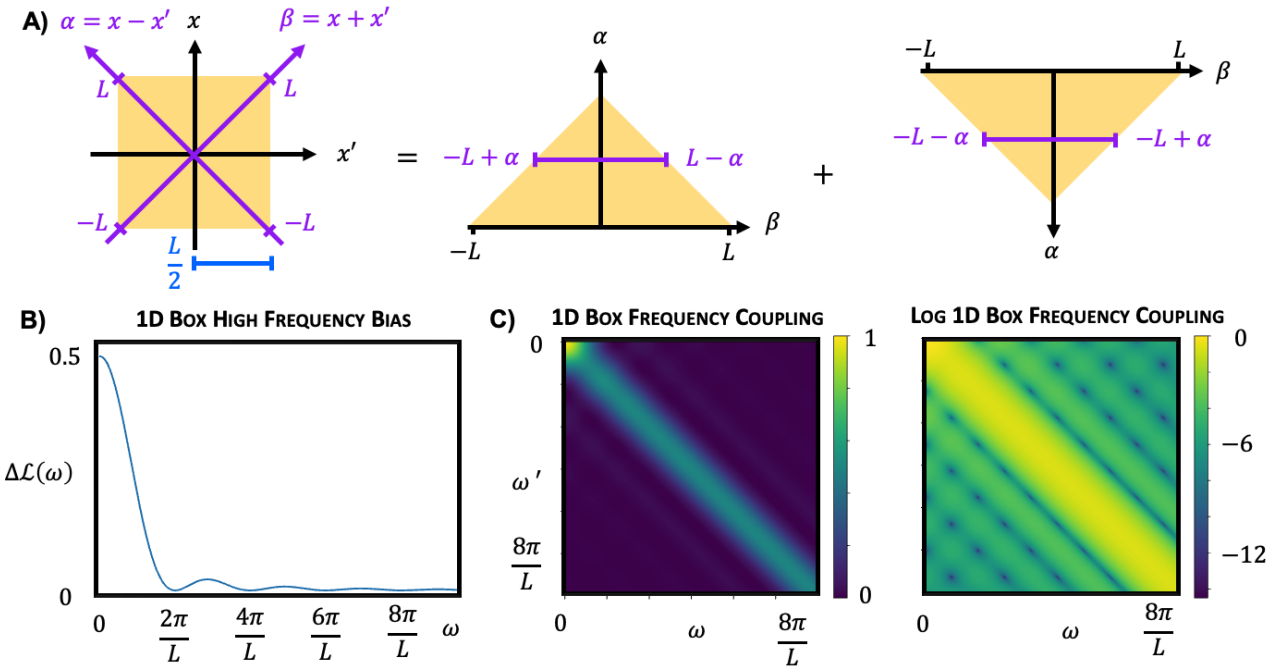


Figure 3.18: **A: Changing Integration Variables** The logic behind the integration change in 3.101. **B: 1D Box Frequency Bias** As in equation 3.104, shows both a high frequency bias and a preference for integer numbers of wavelengths in the box. **C: 1D Box Frequency Interaction** Plot of equation 3.105, shows both a high frequency bias, a repulsion between frequencies, and non-trivial optimal interaction differences.

3.I.2 2-Dimensional Box

A very similar analysis can be applied to the representation of a 2-dimensional box, and used to derive the frequency bias plotted in Figure 5F. We reach an exactly analogous point to equation 3.102,

$$\mathcal{L} = \frac{2}{L^4}\left[\int_0^L \int_0^L e^{-\frac{\sum_d A_d^2 \sin^2(\frac{k_{dx}x + k_{dy}y}{2})}{2\sigma^2}}(L-x)(L-y)dxdy + \right. \tag{3.106}$$

$$\left. \int_0^L \int_{-L}^0 e^{-\frac{\sum_d A_d^2 \sin^2(\frac{k_{dx}x + k_{dy}y}{2})}{2\sigma^2}}(L-x)(L+y)dxdy\right] \tag{3.107}$$

Again we'll study the first order term of the series expansion,

$$\Delta\mathcal{L}_1 \propto \frac{1-\cos(m_d k_{dx} L)}{m^2 k_{dx}^2}\frac{1-\cos(m_d k_{dy} L)}{m^2 k_{dy}^2} \tag{3.108}$$

This is exactly the bias plotted in Figure 3.5F. Changing to rectangles is easy, it is just a rescaling of one axis, which just inversely scales the corresponding frequency axis.

3.I.3 Circular Room

The final analysis we do of this type is the hardest, a circular environment. It follows the same pattern, make the symmetry assumption, expand the series. Doing so we arrive at the following first order term:

$$\Delta\mathcal{L}_1 = \frac{1}{\pi^2 R^4}\iint_0^R \iint_0^{2\pi} \cos(m\boldsymbol{k}_d \cdot (\boldsymbol{x} - \boldsymbol{x}'))d\psi d\psi' r r' dr dr' \tag{3.109}$$

where (r, ψ) is a radial co-ordinate system, $\boldsymbol{x} = \begin{pmatrix} r\cos\psi \\ r\sin\psi \end{pmatrix}$. Using the double angle formula to spread the cosine, and rotating the 0 of each ψ variable to align with $\boldsymbol{k}_d$, we get:

$$\Delta\mathcal{L}_1 = \frac{1}{\pi^2 R^4}\left[\left(\int_0^R \int_0^{2\pi} \cos(mr\|\boldsymbol{k}_d\|\cos\psi)d\psi r dr\right)^2 + \left(\int_0^R \int_0^{2\pi} \sin(mr\|\boldsymbol{k}_d\|\cos\psi)d\psi r dr\right)^2\right] \tag{3.110}$$

The second integral is 0, because sine is odd and periodic. We can perform the first integral using the following Bessel function properties,

$$\int_0^{2\pi} \cos(a\cos(\psi))d\psi = 2\pi J_0(\|a\|) \tag{3.111}$$

Hence,

$$\Delta\mathcal{L}_1 = \frac{4}{R^4}\left(\int_0^R J_0(mr\|\boldsymbol{k}_d\|)rdr\right)^2 \tag{3.112}$$

Which can be solved using another Bessel function property,

$$\int_0^R rJ_0(ar)dr = \frac{RJ_1(aR)}{a} \tag{3.113}$$

Therefore,

$$\Delta\mathcal{L}_1 = \frac{4J_1(m\|\boldsymbol{k}_d\|R)^2}{R^2m^2\|\boldsymbol{k}_d\|^2} \tag{3.114}$$

This is exactly the circular bias plotted in Figure 3.5G. Placing the base frequencies of grid module on the zeros of this bessel function would be optimal, i.e. calling a zero ξ, then the wavelengths of the grid modules should optimally align with,

$$\nu = \frac{2\pi R}{\xi_k} \tag{3.115}$$

This corresponds to grids with wavelengths that are the following multiples of the circle diameter: $0.82, 0.45, 0.31, 0.24, 0.19, .$ Though, as with the other predictions, the major effect is the polynomial decay with frequency, so these effects will all be of larger importance for the larger grid modules. As such, we might expect only the first two grid modules to fit this pattern, the others should, in our framework, focus on choosing their lattice parameters to be optimally non-harmonic.

3.J Representations of Circles, Spheres, and 3-dimensions

We believe our approach is relevant to the representation of many variables beyond 2-dimensional space. Most simply we can directly apply the tools we've developed to the representation of any variable whose transition structure is a group, Section 3.A.1. This is actually true for both numerical and analytic approaches, but here we focus mainly on the numerical approach. We discuss the representation of an angle on a ring, a point on a sphere, and 3 dimensional space. Additional interesting variables might include a position in hyperbolic space (since tree-like category structures, such as the taxonomic hierarchy of species, are best described by hyperbolic spaces), or the full orientation of an object in 3-dimensional space.

For starters, we have frequently referenced the representation of an angle, $\boldsymbol{g}(\theta)$. This is also something measured in brain, in the famous head direction circuit, perhaps the most beautiful circuit in the brain (Kim et al., 2017). There, you measure neurons that have unimodal tuning to heading direction. We match this finding, Figure 3.19A. This is understandable since, as discussed, on a finite space you still want a lattice module, but χ, encourages you to be low frequency and there is no high frequency bias, since the space is finite and $p(\theta)$ is uniform. Thus, place cells are optimal. If we included more neurons it would likely form multiple modules of cells with higher frequency response properties.

We can also apply our work to the representation of a point on a sphere. Again, with the low frequency bias place cells are optimal, Figure 3.19B. Without it, or conceivably with enough neurons that multiple modules form, you can get many interesting patterns, Figure 3.19C-D. These could be relevant to the representation of object orientations, or in virtual reality experiments where animals must truly explore spherical spaces.

Finally, we can equally apply our framework to the representation of a point in three dimensional space. Grid cells have been measured while an animal explores in 3D and shown to have irregular firing fields in both bats (Ginosar et al., 2021), and mice (Grieves et al., 2021). Analogously to the griddy modules in one and two dimensions, our framework predicts griddy three dimensional responses that are similarly densely packing, which is verified in 3.19E. However, in Appendix 3.F we performed ablation studies and found that removing the actionable constraint led to multimodal tuning curves without structure, and we find the same for 3D, Figure 3.19F, perhaps matching the multimodal but unstructured representation found in the brain (e.g. Figure 3.19G from Ginosar et al., 2021). As such, it seems that in three dimensions the animals are building a non-negative and discriminative representation without the constraint to path integrate. Therefore, we suggest that the animals have sacrificed the ability to manipulate their internal representations in 3D and instead focus on efficiently encoding the space.

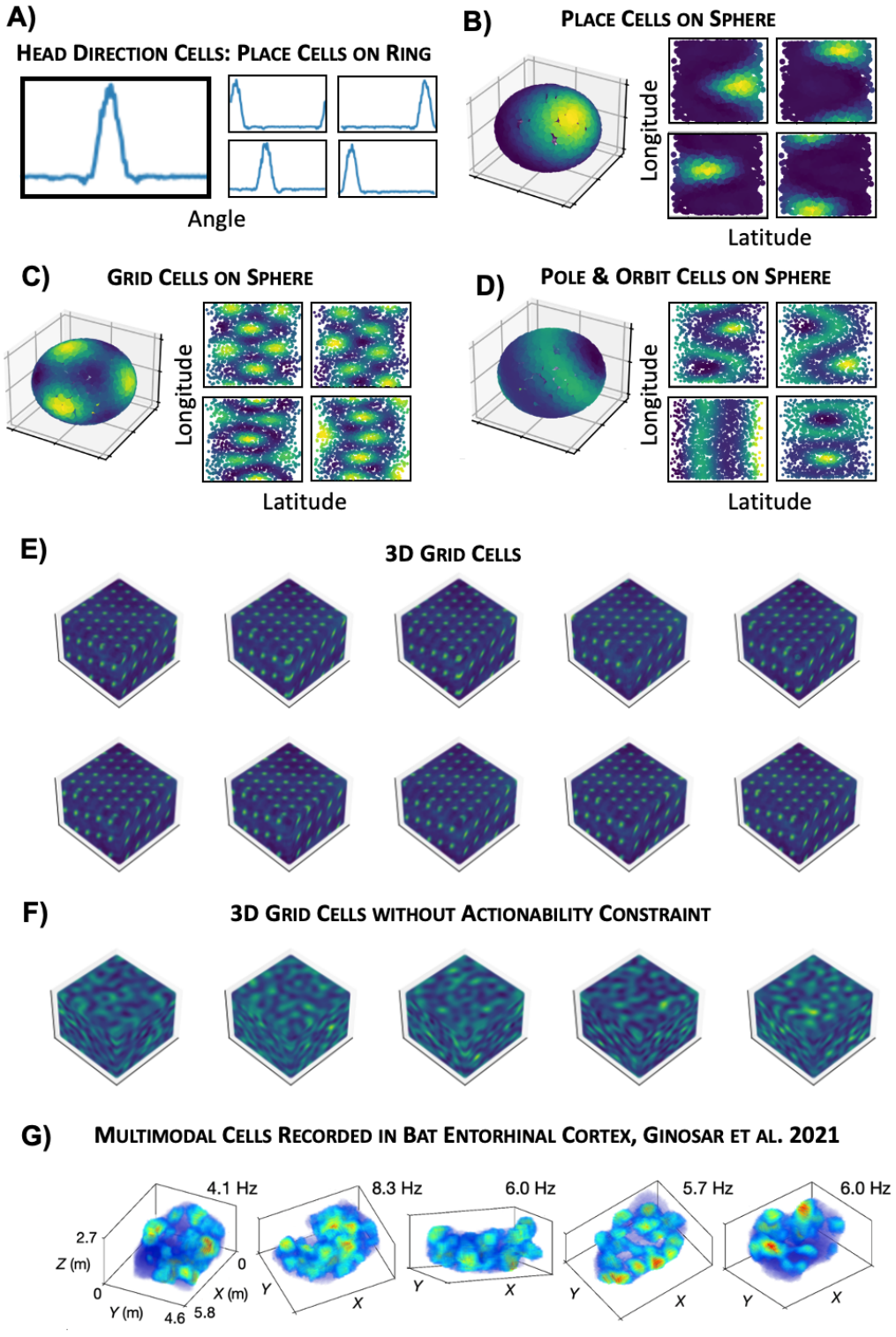


Figure 3.19: **A** Representation of an angle on the ring. With a low frequency bias the optimal one module response is a set of head-direction cells. **B** Representation of position on a sphere. With a low frequency response place cells are again optimal. **C-D** Allowing the optimiser to find other modules produces funky looking responses on the sphere, like grids (**C**) and 'pole & orbit' cells (**D**.) **E** A module of densely packing grids in 3D space. **F** The same cells without the actionability constraint, as we did for 2D in Figure 3.15. **G** Recordings of multimodal cells from bat entorhinal cortex from Ginosar et al., 2021.

Chapter 4

ANOTHER NORMATIVE GRID CELL THEORY?

For 20 years the beautiful structure in the grid cell code has presented an attractive puzzle: what computation do these representations subserve, and why does it manifest so curiously in neurons. The first question quickly attracted an answer: grid cells subserve path-integration, the ability to keep track of one's position as you move about the world. Subsequent work has only solidified this link: bottom-up mechanistic models that perform path-integration match the measured neural responses, while experimental perturbations that selectively disrupt grid cell activity impair performance on path-integration dependent tasks. A more controversial area of work has been top-down normative modelling: why has the brain chosen to compute like this? Floods of ink have been spilt attempting to build a precise link between the population's objective and the measured implementation. The holy grail is a normative link with broad predictive power which generalises to other neural systems. We review this literature and argue that, despite some controversies, the literature largely agrees that grid cells can be explained as a (1) biologically plausible (2) high fidelity, non-linearly decodable code for position that (3) subserves path-integration. As a rare area of neuroscience with mature theoretical and experimental work, this story holds lessons for normative theories of neural computations, and on the risks and rewards of integrating task-optimised neural networks into such theorising.

4.1 Introduction

It has been 20 years since the discovery of the most surprising single neuron response yet described: grid cell activity correlates with an animal's self-position, activating when the animal is in a hexagonal lattice of positions (Hafting et al., 2005), fig. 4.1. Perhaps even more surprising than their original discovery is the finding that the grid cells lattices come in discrete modules of which a single rodent will have a handful (H. Stensola et al., 2012). Grid cells in the same module have receptive fields that are translated (but not rotated) versions of one another which uniformly tile the space of possible phases, fig. 4.1. Finally, alongside the grid cells in layer II of medial entorhinal cortex, layer III hosts cells that fire at conjunctions of a hexagonal lattice of positions and a particular heading direction (Sargolini et al., 2006). There exists extensive additional phenomenology; but these four fundamental features form a cohesive explanatory target:

1. Hexagonal-lattice tuning curves
2. All grid cells belong to a module, a group of many grid cells whose tuning curves are translated versions of one another, tiling the phases uniformly.
3. The entire grid cell code comprises a handful of modules with different lattices, each comprising many neurons.
4. The existence of paired conjunctive grid-heading direction cells.

Giving these striking findings our questions are clear: what do grid cells do? And why in this way?

A large body of work has convincingly answered the first question: the grid cell representation subserves path-integration. It has long been posited that the mammalian brain is capable of integrating its velocity to track self-position (Tolman, 1948), and as soon as grid cells were discovered they became the likely neural implementation (McNaughton et al., 2006). In the intervening time the evidence has only built. We now briefly review two of the key strands of evidence for this, mechanistic models and perturbation effects.

On the one hand, mechanistic models that perform path-integration match neural observations. The predominant modelling approach is continuous attractor circuits, which were initially developed to model

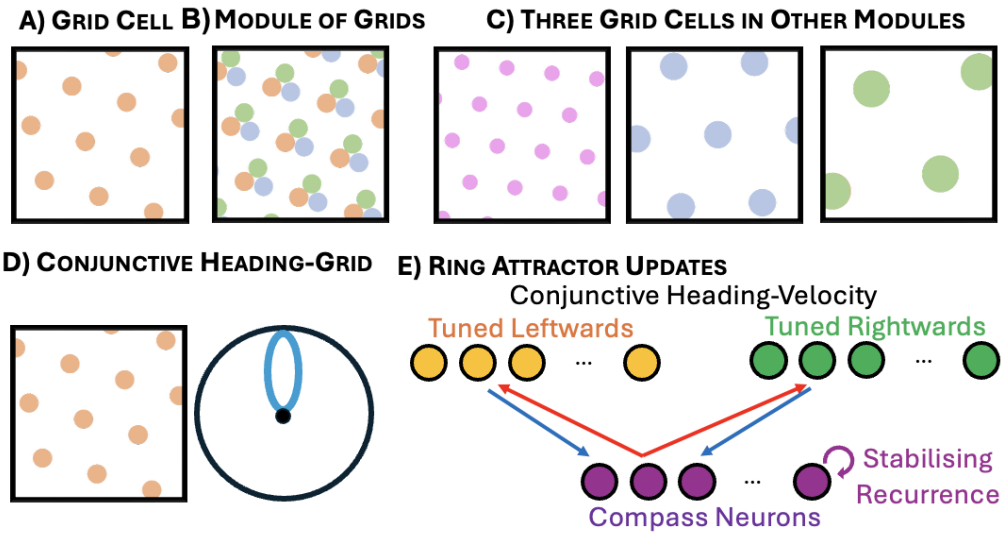


Figure 4.1: **Structure of the grid cell code. A:** Neurons are tuned to a hexagonal lattice of positions in 2D space. **B:** They are grouped into modules: neurons in the same module have translated (but not rotated) receptive fields, and across a module they uniformly sample the phases (translations). **C:** There are only a handful of modules in one animal, each with its own lattice, and 100s of neurons covering the possible phases. **D:** For each grid module there is a population of grid cells that are conjunctively tuned to both the underlying grid of the module, and a particular heading direction. **E:** These conjunctive neurons can implement path-integration by pushing the bump of activity around the module (Burak and I. R. Fiete, 2009), like the ring attractor in the fly central complex (Hulse and Jayaraman, 2020), using a shifted connectivity pattern: conjunctive neurons project to the pure grid cells in the same module with a receptive field that is translated in the direction of the conjunctive neuron's heading tuning (Vollan et al., 2025), exactly as required for path-integration.

path-integration of heading direction. Their simplest implementations comprise one population of neurons that encode the animal's heading direction, then two further populations that code for conjunctions of heading direction and angular velocity either to the left or right. These conjunctive heading-velocity neurons can then be used to update the heading direction representation, fig. 4.1. First theoretically posited in the 90s (Redish, Elga, and Touretzky, 1996; Skaggs et al., 1994), these circuits have since been verified experimentally, most beautifully in the fruit fly (Kim et al., 2017). Subsequent work extended these to two-dimensional space to model hippocampal place cells (Conklin and Eliasmith, 2005; Samsonovich and McNaughton, 1997; Touretzky and Redish, 1996). One difficulty in moving from a finite space of heading directions to an infinite space of (2D) positions is encoding the space in a finite set of neurons. Work that predated the discovery of grid cells proposed encoding space periodically, predicting lattice tuning curves but with square rather than hexagonal lattices (Samsonovich and McNaughton, 1997). Subsequent work has shown how attractor dynamics in these 2D continuous attractor circuits can naturally leads to hexagonal grid and conjunctive cells (Burak and I. R. Fiete, 2009), and multiple modules (L. Kang and Balasubramanian, 2019; Khona, Chandra, and I. Fiete, 2025). As such, it seems natural to conceptualise each module of the grid cell system as a continuous attractor network capable of path-integrating. There are other mechanistic models of grid cells that perform path-integration, notably the oscillatory-inteference model (Burgess, Barry, and O'keefe, 2007), which are reviewed elsewhere (Giocomo, M.-B. Moser, and E. I. Moser, 2011). For current purposes the diversity of models is not an issue, all successful models perform path-integration, supporting the mechanistic evidence that grid cells can subserve path-integration.

Concurrently, behavioural evidence has shown that perturbing the grid cell system impairs animals' ability to perform path-integration dependent tasks. First, lesions to the medial entorhinal cortex impair path-integration (Steffenach et al., 2005; Van Cauter et al., 2013). Second, disrupted spatial navigation is a known symptom of Alzheimer's disease, and this effect is thought to arise due to disruptions in grid coding in the medial entorhinal cortex. Evidence comes from genetic knock-in models of Alzheimer's which have disrupted grid cells (Jun et al., 2020; Ying et al., 2022), alongside impaired path-integration abilities (Ying et al., 2022). Further, people at genetic risk of Alzheimer's show disrupted grid coding long before displaying other symptoms of Alzheimer's (Kunz et al., 2015). Finally, and most precisely, removal of NMDA glutamate receptors from retro-hippocampal regions led to a selective disruption of grid cells while leaving other spatially selective cells intact. This perturbation caused behavioural disruptions to path integration (Gil et al., 2018). In sum, the behavioural evidence is specific and strong.

Parallel to these efforts, normative work has attempted to understand why the brain has chosen to implement

the computation in this particular way. Normative approaches propose an objective: a teleological goal of the representation measuring its quality. Then one searches for the optimal choice, subject to various constraints. The optimal representation is then compared to (and hoped to be predictive of) the brain. These optimisations can either be analytic, or numerical, with extensive recent work using task-optimised neural networks to find the putatively optimal solution. These normative approaches have a long pedigree in neuroscience, mainly in sensory representations where the efficient coding hypothesis (Attneave, 1954; Barlow et al., 1961) has been repeatedly used to understand puzzling neural responses, such as tuning curve properties (Laughlin, 1981; Olshausen and Field, 1996). These approaches are attractive, allowing justification of what are otherwise often hand-selected features of mechanistic models, and highlighting the core principles at work, permitting their generalisation to analogous problems.

Given the overwhelming body of careful evidence, the normative question seems well-posed and tractable: why has the brain chosen to path-integrate in this way, and what broader conclusions can we draw for neural coding and computation. Yet, there has been significant controversy in the field, with many proposed models ignoring path-integration entirely. Here we review this extensive literature. We begin with 'efficient encoding' models that don't have a notion of path-integration. These models have certainly been useful. However we argue, among other things, that the axis-aligned nature of a grid cell module is hard to justify without path-integration, and find that almost all such approaches are unable to do so. Then in section 4.3 we review models which include path-integration. We argue that, despite the controversy, multiple models fit all the key properties using a core set of ideas: a (1) path-integrable (2) biologically plausible, (3) high fidelity, non-linearly code for position. The sharing of these ideas despite implementational differences suggest they are relatively robust. Finally, in serving as a well-characterised whetstone for normative theorising, this endeavour holds broader lessons that we discuss in section 4.4, before concluding with open problems in the world of grid cell theory.

4.2 Grid Cells are not *just* an Efficient Code for Space

We begin by reviewing a large body of normative work that does not include any notion of path-integration. Instead, these approaches take what we term an 'efficient coding' approach: first you propose some variables to be encoded, then you devise one objective or constraint that captures the encoding quality, another the efficiency (e.g. number of spikes). Finally, you optimise the encoding under an energy constraint, or minimise the energy under an encoding constraint etc.(Laughlin, 1981; Olshausen and Field, 1996). Grid cells are clearly encoding self-position making the first step (choosing a variable) easy; the methods then differ in their choice of encoding or energy term. We argue that the lack of path-integration explains why no efficient coding approach is able to capture the entire behaviour of the grid cell code, though many present useful ideas that will recur in the fuller theories presented later.

4.2.1 Context: Often Grid Cells are not the most efficient code for Space

Before proceeding we make a useful counterpoint: most natural instantiations of efficient coding do not produce grids. By comparing to these theories the choices that led to grids become clearer. For example, (Sengupta et al., 2018) use the similarity matching objective: given two inputs, $\boldsymbol{x}$ and $\boldsymbol{x}'$, and their neural encodings, $\boldsymbol{z}(\boldsymbol{x})$ and $\boldsymbol{z}(\boldsymbol{x}')$, this objective encourages the dot-product similarity of the representation, $\boldsymbol{z}(\boldsymbol{x})^T\boldsymbol{z}(\boldsymbol{x}')$, to match that of the input similarity structure, $\boldsymbol{x}^T\boldsymbol{x}$, through maximising the following loss:

$$\mathcal{L} = \iint (\boldsymbol{x}^T\boldsymbol{x}' - \alpha)\boldsymbol{z}(\boldsymbol{x})^T\boldsymbol{z}(\boldsymbol{x}')dp(\boldsymbol{x})dp(\boldsymbol{x}') \tag{4.1}$$

Sengupta et al., 2018 take inputs from a compact continuous space, such as angles on a ring, and (reasonably) assume that the input similarity, $\boldsymbol{x}^T\boldsymbol{x}$, decays with distance: nearby points are similar, distant are dissimilar. From this they analytically derive that, with infinitely many neurons, place cells are the optimal nonnegative representation. This somewhat natural conclusion is not specific to this loss: recent work has drawn similar conclusions from an information theoretic measure of coding quality (Deighton et al., 2024). This should not be too surprising, place cells are a very informative code, and a much simpler one than grid cells. When there are enough neurons such that a place cell code can tile the space with sufficient resolution, most reasonable efficient coding losses prefer place cells.

This leaves us with a few routes to derive grid cells. The first is easiest. One can assume that grids are optimal, and then use efficient coding approaches to derive the grid's properties. This is a very sensible and informative approach which we will review in section 4.2.5, but it will not explain when grids are optimal. A second option is to change the problem setup. There are various such morphed efficient coding approaches that derive grid cells which we review in the next few sections. However, we highlight that the most natural versions of the problem do not lead to grid cells, making the changes required particularly intriguing.

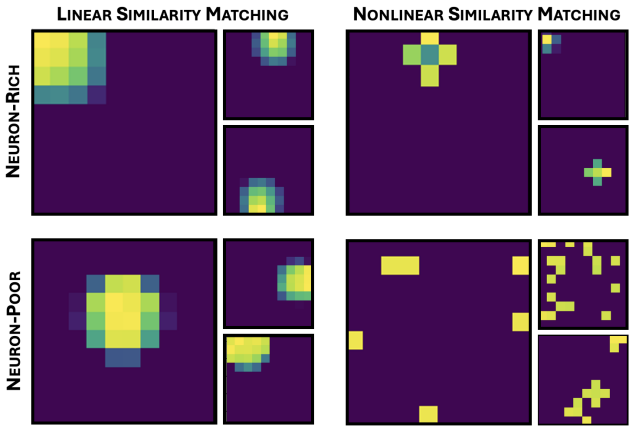


Figure 4.2: We create a neural representation of a set of 2D positions and measure the similarity between the representation of all pairs of points. We optimise the representation so that it is nonnegative, and such that the neural similarity is sharply peaked and decays with distance. We either measure similarity linearly, left column, using the dot product the of the neural encodings of each position $\boldsymbol{z}(x)$: $\boldsymbol{z}(\boldsymbol{x})^T\boldsymbol{z}(\boldsymbol{x}')$; or nonlinearly, right column, by measuring the exponential of the neural distance between two points ($e^{-\frac{||\boldsymbol{z}(\boldsymbol{x})-\boldsymbol{z}(\boldsymbol{x}')||_2^2}{2\sigma^2}}$). We plot the rate maps across space of three of the optimal neural responses for each loss with either many (top row) or few neurons (bottom row). When there are enough neurons to tile the space at the required resolution, place cells are optimal (neuron-rich, top row). When there are few neurons place cells are still optimal for a linear solution, but multifield (but not griddy) cells are optimal for the nonlinear similarity measure, bottom row.

A final option follows the observation that the optimality of place cells relies on there being enough neurons such that place cells could tile the environment at a sufficient resolution. Perhaps grid cells are simply the optimal efficient code when there are fewer neurons? We find that this is also not the case. We optimise a nonnegative code subject to unit-norm firing rate constraint to either a linear similarity, or nonlinear similarity metric fig. 4.2. As claimed, when the number of cells is large place cells are ubiquitously preferred. On the other hand, with a small number of neurons we see a separation of cases. If the similarity measure is linear (corresponding to a weak position decoder) place cells are still optimal, whereas if the similarity measure is nonlinear (corresponding to a flexible nonlinear decoder) the optimal representation are is multimodal, like place cell recordings in large environments (Eliav et al., 2021; Fenton et al., 2008; Harland et al., 2021; J. S. Lee et al., 2020; Rich, Liaw, and A. K. Lee, 2014; Yartsev and Ulanovsky, 2013). They are never form an axis-aligned module of grids.

On further reflection this is not very surprising. The grid-cell code has some glaring design flaws from a pure efficient coding perspective. Grid cells are roughly periodic, meaning they identically encode points separated by the lattice symmetry, rendering the cell unable to distinguish them. More importantly, grid cells come in modules: cells within the same module have the same lattice symmetry. This means that rather than helping each other to decode new points, points that are indistinguishable to one neuron in the module are also indistinguishable to all neurons in the same module[1]! Breaking the symmetry, either by rotating and scaling the grid lattices of different neurons or removing the lattice entirely, breaks these symmetries improving the potential performance of the code. We will see that aligning the grid cells within one module makes sense for path-integration: shared symmetry allows you to predict the impact of actions. But it makes little sense from an efficient coding perspective. Correspondingly, we will find that most pure efficient coding approach are unable to explain this piece of the grid code structure.

We now review the various efficient coding approaches to grid cells. In particular, given this framing, we review choices that each makes in order to make grid cells emerge from a problem that, when framed in most natural ways, does not produce grid cells.

4.2.2 Idea I: Dense Packing

Hexagonal lattices are the densest packing of spheres in 2D space, or analogously, the best arrangement of sensors to minimise the average distance between all points in 2D space and the nearest sensor. One family of approaches use this idea to produce hexagonally tuned cells.

[1]There are some ways round this, for example each grid cell in one module might be slightly rotated relative to one another (Redman et al., 2025) allowing one module to decode all points. This is an interesting finding, but normatively it remains unclear why in an efficient-coding only perspective something this limited was done though: why not completely randomise the angles? Why align them at all?

Mok and Love, 2019 argue that place cells form a conceptual clustering of inputs: which place cells is active for each input corresponds to its cluster and the quality of the encoding is given by the resolution of the clustering[2]. They argue that space can be thought of as a uniform continuum of inputs to be explained, and that, thanks to dense packing, the optimal choice of a finite set of place cells (clusters) is a hexagonal grid. They then argue that grid cells are a measure of proximity between points in space and their nearest cluster - which in this model is a measure of how well fit that point is by the learnt clusters. Since the data is best explained at cluster centres this forms a hexagonal lattice.

Ginosar et al., 2021, prompted by their discovery of non-periodic encodings by grid cells of three-dimensional space (a topic we'll return to), present a parsimonious model that explains both 3D and 2D representations. They model grid fields as particles that repulse each other at short distances and attract at intermediate, the dynamics then pushes the particles towards lower energy states, and the optimal state is a dense packing. Matching neural observations, running these dynamics in 2D leads to dense packing hexagonal lattices, while in 3D it often leads to jammed sub-optimal solutions without global periodic structure.

A slightly related idea appears in Huber, 2025. In this memory model the classic roles of place and grid cells are reversed, place cells encode where a memory happens (a conjunction of a thing and a place) while grid cells encode the thing that is happening. Grid tuning curves are produced by arguing that the grid cell is encoding a variable that is uniform across space. The model then assumes that inputs that are nearby in space will be grouped into the same memory, while those beyond a critical distance will trigger a new memory. These dynamics lead to a hexagonal lattice receptive field, which can be understood via dense packing.

Despite the elegant simplicity of these approaches, simple functional questions remain non-obvious and key phenomena unexplained. No approach naturally incorporates the axis-aligned nature of cells within one grid module: in Mok & Love or Huber it is not obvious why grid cells would code for a translated version of either the conceptual fit to data or a set of memories, while in Ginosar et al. some mechanism would be required to align these densely packing lattices across neurons. Similarly unclear is why there are modules with a discrete set of lengthscales or conjunctive grid cells. Finally, why we should think of grid cells as a measure of hippocampal fit, as a discretised version of a uniform variable, or as a set of repulsing particles, when more compelling narratives exist is unclear. Nonetheless, in conjunction with other ideas, dense packing does explain the choice of hexagonal lattice in many models (Will Dorrell et al., 2023; M. Stemmler, Mathis, and Herz, 2015).

4.2.3 Idea II: Nonnegative Encoding of Fourier Features

The second set of grid cell theories use an encoding objective that rewards the representation for containing high power at a critical spatial frequency, then use nonnegativity to form a hexagonal lattice. The pivotal link in these arguments was first described by Dordek et al. (2016): they performed principal components analysis (i.e. constructed the optimal linearly-decodable encoding) of difference-of-Gaussian place cells and found the optimal representation was square grid cells. Additionally constraining the representation to be nonnegative led to hexagonal grid cells. The similarities of this approach to those in section 4.2.1 are large, the only difference is the choice of target: rather than Gaussian place cells, whose similarity structure decays with distance, they use difference-of-Gaussian cells. Dordek et al. (2016) (later paralleled by Sorscher, G. Mel, et al. (2019) and Sorscher, G. C. Mel, Ocko, et al. (2023)) explain the effect of this substitution: difference-of-Gaussian cells lead to a covariance structure peaked at a particular frequency band, leading the optimal linearly-decodable representation to highly encode this frequency. Combining this with a lattice discretisation effect from the finite room leads to square grid cells (Dordek et al., 2016). Finally, positivity changes the optimal solution from square to hexagonal grids, justified either through a triplet interaction effect (Sorscher, G. Mel, et al., 2019; Sorscher, G. C. Mel, Ocko, et al., 2023), or the efficiency in positivising the code (Dordek et al., 2016).

This approach has been influential with many papers using the nonnegative PCA of DoG place cells (Dordek et al., 2016; V. Schøyen et al., 2023; Sorscher, G. Mel, et al., 2019; Sorscher, G. C. Mel, Ocko, et al., 2023; M. Tang, Barron, and Bogacz, 2024). It has also been controversial, prompting a rebuttal (Schaeffer, Khona, and I. Fiete, 2022), a rebuttal to the rebuttal (Sorscher, G. C. Mel, Nayebi, et al., 2022), and two further rebuttals cubed (Schaeffer, Khona, Bertagnoli, et al., 2023; Schaeffer, Khona, Koyejo, et al., 2023). One point of disagreement lay in the finetuning of parameters required to produce grid cells: an interesting point, but clearly not fatal since the brain could simply use these parameters. A more existential threat comes from the choice of difference-of-Gaussian tuning curves. These fit hippocampal place cells less well than Gaussian curves, but, as the theoretical analysis states, are clearly vital for the production of hexagonal grid cells. Many more realistic choices of place cells don't produce grid cells in this framework (Schaeffer, Khona, Koyejo, et al., 2023). This could be an interesting prediction about the relationship between place and grid coding, but currently there's no evidence this particular link exists.

Further, these approaches do not produce axis-aligned grid cells. In fact, they produce grid cells whose orientations cluster into two groups offset at 30 degrees, a pattern that is not observed experimentally. Further,

[2] i.e. The best clustering would give every input its own cluster, the worst would assign all inputs to one cluster.

they do not produce multiple meaningful modules; those that do emerge appear to be related by constants reflecting the numerical discretisation of the space rather than true parts of the problem (Sorscher, G. Mel, et al., 2019), nor does the framework offer an explanation of conjunctive cells. Only when combined with a path-integrating task (for example by training an RNN to both path-integrate and linearly project to DoG place cells) do you get axis-aligned grid cells, a topic we'll return to.

One final related view is presented by K. L. Stachenfeld, Botvinick, and Gershman (2017). This work argues that the hippocampus encodes a successor representation of space, and that grid cells form a nonnegative eigendecomposition of this successor representation. The eigenvectors of a diffusive transition matrix, and hence of a successor matrix, are lattices (a point also raised by (C. Yu, Behrens, and Burgess, 2020)), allowing them to be used as models of grid cells. However, it is tough to use these as predictive models of grid cells. The cells come in modules of two neurons, many of which are not hexagonal grids but instead form bands or amorphous blobs especially in non-square rooms. Further, one of the selling points of this theory is its sensitivity to the transition statistics of the animal: pure grids emerge only with a diffusive policy. This makes a lot of interesting predictions that we will return to it in the discussion, but even in the simplest diffusive setup the theory is lacking. One would have to explain why only some of the eigenvectors match grid behaviour, why grid cells have more than 2 neurons per module that uniformly tile the phase, to have a more meaningful account of multiple modules, and to explain the need for conjunctive grid cells. As yet there has been no reported success in doing this.

4.2.4 Idea III: Metric Encoding

A metric is a function that measures distances between points. Grid cells have been posited to encode space in a way that preserves its metric, meaning that the distance between two points, $\boldsymbol{x}$ and $\Delta\boldsymbol{x}$, is preserved in the representation of those points, $\boldsymbol{z}(\boldsymbol{x})$ and $\boldsymbol{z}(\Delta\boldsymbol{x})$, at least for a small region of space (small $\Delta\boldsymbol{x}$):

$$\boldsymbol{z}(\boldsymbol{x}+\Delta\boldsymbol{x})-\boldsymbol{z}=s||\Delta\boldsymbol{x}||+\mathcal{O}(\Delta\boldsymbol{x}^2)$$

where s is a scaling factor. Normative approaches including losses like these are common routes to grid cells(Gao, J. Xie, Wei, et al., 2021; Gao, J. Xie, Zhu, et al., 2019; Pettersen et al., 2024; D. Xu et al., 2025). We'll return to the results that include path-integration later, but in this section we focus on Pettersen et al., 2024, who, among other things, find that simply optimising a nonnegative representation to preserve distances while penalising the L1 norm of the firing rates is sufficient to generate hexagonal firing fields. The loss used is:

$$\mathcal{L}=\underbrace{\alpha\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[e^{-\frac{1}{2\sigma^2}||\boldsymbol{x}-\boldsymbol{x}'||_2^2}(||\boldsymbol{x}-\boldsymbol{x}'||_2-||\boldsymbol{z}(\boldsymbol{x})-\boldsymbol{z}(\boldsymbol{x}')||_2)^2\right]}_{\text{Conformal Isometry}\approx\text{Modified Similarity Matching}}+\underbrace{(1-\alpha)\mathbb{E}[||\boldsymbol{z}(\boldsymbol{x})||_1]}_{\text{L1 Capacity Loss}} \tag{4.2}$$

We'll understand this by showing that the loss performs basically the same fourier selection procedure as the difference-of-gaussian place cells. We can see that this conformal loss compares the similarity of the input points, $||\boldsymbol{x}-\boldsymbol{x}'||_2$, to that of the output points, $||\boldsymbol{z}(\boldsymbol{x})-\boldsymbol{z}(\boldsymbol{x}')||_2$, and minimises the difference. In this sense, the objective is a form of similarity matching as in Sengupta et al., 2018, with the minimisation weighted by $e^{-\frac{1}{2\sigma^2}||\boldsymbol{x}-\boldsymbol{x}'||_2^2}$. This weighting sets a lengthscale on the local region in which similarity matching has to occur. Indeed, if σ is much larger than the environment, $e^{-\frac{1}{2\sigma^2}||\boldsymbol{x}-\boldsymbol{x}'||_2^2}\approx 1$, and the loss becomes a similarity matching one, and place cells will likely be optimal according to the results Sengupta et al., 2018.

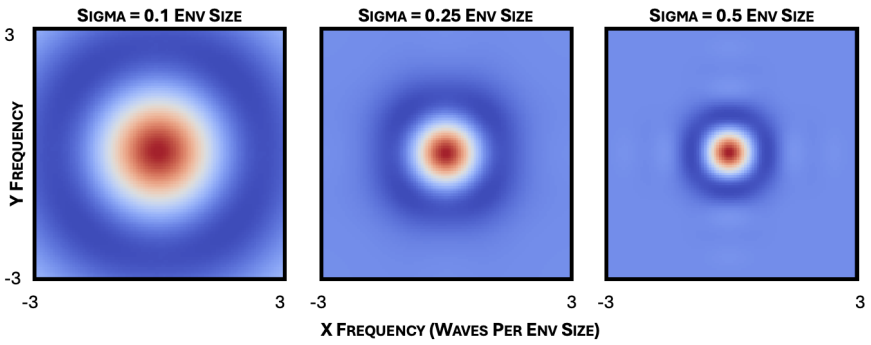


Figure 4.3: The loss is biased in fourier space towards a particular peak frequency, as in difference-of-gaussian losses, that controls the lengthscale of the grid cells.

What about smaller σ? This loss can be analysed in fourier space, and seen to produce a similar effect to difference-of-gaussian place cells from the previous section. The local region, encapsulated by σ, sets a lower

bound on the frequency content of the code: if your code contains a component oscillating at slower than $\sim \frac{1}{\sigma}$ it won't have varied meaningfully within the regions you care about, so won't decrease the loss. Conversely the similarity matching part, $(||\boldsymbol{x} - \boldsymbol{x}'||_2 - ||\boldsymbol{z}(\boldsymbol{x}) - \boldsymbol{z}(\boldsymbol{x}')||_2)^2$, is effectively a low-frequency bias: the code should vary so that nearby points are similar, and distant ones are different. To illustrate this we can assume the code is made from a single frequency and plot the loss as a function of this frequency fig. 4.3, we can see the loss is minimised at a particular frequency that scales with the inverse of σ.

Similar arguments to the previous section can then be used to justify how positivity and capacity constraints might lead to grid cells. Indeed, hexagonal grid cells with a single lengthscale emerge from this optimisation, and the largescale is controlled by σ. This is not a complete picture: for example, it is an interesting mathematical puzzle that combining this loss with an L1 capacity constraint, but not an L2, leads to hexagonal grids (Pettersen et al., 2024). Regardless, these grid cells still suffer from the same shortcoming of other efficient coding only approaches: the grids are not aligned within the same module, they feature the same loose 30° alignment as the Fourier approaches. Only by adding path-integration is this effect removed.

4.2.5 Idea IV: Optimising a Multi-Modular Axis-Aligned Grid Code

A final efficient coding approach has had much more success at matching detailed features of the code. This approach assumes the representation is comprised of multiple modules of axis-aligned grid cells, and then make statements about the optimal choices of grid lattices.

Early results demonstrated that a multi-modular grid cell code could be used to accurately decode position to high accuracy, higher than a place cell code (Mathis, Herz, and M. Stemmler, 2012; Sreenivasan and I. Fiete, 2011). Various approaches then studied the optimal choices of grid lengthscales or lattice symmetry to maximise the fisher information (Mathis, Herz, and M. Stemmler, 2012; Mathis, Herz, and M. B. Stemmler, 2012), or decoding accuracy with various decoders (M. Stemmler, Mathis, and Herz, 2015; Wei, Prentice, and Balasubramanian, 2015), with some emphasising the effect of independent per-module noise (Towse et al., 2014). These approaches have been useful in understanding the properties of the grid cell code. For example, works have used these ideas to predict a relationship between grid scale and the number of grid cells in each module (Mosheiff et al., 2017). Similarly, a lot of work has analysed the optimal choice of ratio between the lattice lengthscales of successive grid modules. Multiple methods appeared to match early experimental results (H. Stensola et al., 2012), however, as yet, the experimental truth remains slightly unclear, potentially due to the difficulty in measuring many modules simultaneously. Further, recent models have posited completely different justifications of the lengthscale ratios based on developmental arguments (Khona, Chandra, and I. Fiete, 2025). Despite this unfinished story, the proximity to predicting testable phenomena is appreciated.

Of course, assuming a multi-modular and axis-aligned grid code cannot tell us why such things emerge. One set of these models that does make such statements are those that that assume the decoder is a population vector decoder (Dietrich et al., 2024; M. Stemmler, Mathis, and Herz, 2015). The simplest version of a population vector decoder has long been used for heading direction (Salinas and Abbott, 1994). Each cell is unimodally tuned to a particular heading: θ_i, and fires with a firing rate g_i. An estimate of the angle is calculated by summing the sine and the cosine of the maximal directions weighted by the firing rates:

$$\text{e.g. } \cos(\hat{\theta}) = \frac{1}{N}\sum_i g_i \cos(\theta_i) \tag{4.3}$$

Under various conditions these estimators are optimal (M. Stemmler, Mathis, and Herz, 2015), and these decoders can also be used to decode position from a module of grid cells up to the lattice symmetry of the grid. One normative justification of the grid systems is that the modular-aligned structure allows optimal performance given such a decoder (M. Stemmler, Mathis, and Herz, 2015). However, the structure of this decoder requires assuming that each neuron is encoding one particular position up to a symmetry (like the heading θ_i), and so it is built in such a way that ensures a grid-like code will be optimal. Further, taking full advantage of a multimodular code requires a nonlinear decoder: combining many linear decoders can only permit more accurate decoding within the grid scale of the largest grid module, rather than permitting decoding of a range much larger than the largest grid scale. Finally, the existence of unstructured multifield representations elsewhere in the brain suggests that the nervous system can implement more flexible decoders. As such, this is a reasonable hypothesis, but one that bakes in the modular structure through the limitations of the decoder, that cannot on its own explain the existence of conjunctive grid-velocity cells, that predicts that accurate decoding only within the range of the largest grid module, and that I find less convincing than its combination with path-integration and a more flexible decoder.

4.2.6 Efficient Coding Conclusions

From this large body of work I conclude that grid cells, despite clearly being a good code, are not the optimal efficient code of 2D space in most natural instantiations of the problem. Under many versions of the problem,

simply optimising a neural representation leads to place cells when there are enough neurons to make such a solution effective, or multifield but non-griddy cells when there are too few neurons, section 4.2.1. This matches unpublished findings from Tzushuan Ma's PhD thesis (Ma, 2020), and recent work that has shown such multifield place cells in the hippocampus when the environment is large (Eliav et al., 2021; Harland et al., 2021; Rich, Liaw, and A. K. Lee, 2014). Changing the problem in various ways can make the grid cells solution optimal. This is most often done by encouraging a particular frequency band to be present in the code, either through read-out of difference-of-gaussian place cells section 4.2.3, or a metric approach section 4.2.4. Almost all of these approaches are unable to explain the axis-alignment of grid cells, which is deleterious from a pure efficient coding perspective. Normative approaches that assume a population vector decoding approach are able to explain much of the grid cell code, though not the existence of conjunctive neurons, and rely on a fairly limiting decoding assumption. We now move back to path-integration based approaches.

4.3 Grid Cells form a Path-Integrable Representation of Space

Each module of grid cells comprises a large population of neurons with translated receptive fields. This is perfect for path-integration. Let's say in your current position one neuron is firing. Due to the lattice symmetry, you only know where you are up to the degeneracy of this neuron's firing. However, thanks to the lattice symmetry, no matter which of this neuron's grid fields you are in, you know that when you step right the next neuron that should fire is the neuron whose receptive field lattice is translated one step to the right. Therefore this code, that is suboptimal for encoding positions, is excellent at predicting the consequences of your actions: it permits forward modelling of your self-position. This elegant idea has been clear from the earliest theories (McNaughton et al., 2006; Samsonovich and McNaughton, 1997), and has appeared in a plethora of work since, that together fit many aspects of the grid cell code. Further, implementing this in neurons leads naturally leads to conjunctive grid-head-direction neurons. When you take a step you need to push the bump along the attractor manifold; just like in a ring attractor (or in linear look-ahead models (Kubie and Fenton, 2012)) the ideal neurons to perform this are the conjunctive heading-grid neurons. We now review a few approaches to turning this into a normative theory.

Efficient Coding of Trajectories Rebecca et al. (2025), following similar work by Waniek (2020), study sequence coding: the idea that each pair of sequentially active grid representations uniquely identifies the movement direction. This is like a reversed framing: rather than saying that from your current encoding and a velocity you can predict your next encoding, they say that velocity can be predicted from the pair of current and next encoding. From this coding scheme and a small number of assumptions they show that a single hexagonal grid module is optimal. This argument suffers from using binary neurons and a discretisation of space, and struggles to naturally encapsulate multiple modules. Despite this, it is an elegant alternative formulation of the idea of path-integration, with some novel predictions, such as the way in which a 2D module should encode a 1D sequence.

Path-Integrable Efficient Codes A second approach, followed by us (Will Dorrell et al., 2023) and similar to unpublished work (Ma, 2020), bears more resemblance to the earlier efficient coding approaches. Identically to an efficient coding approach, the representation is asked to encode space, subject to some efficiency constraints. However, crucially, the code is also asked to permit path-integration. This constraint is enforced using action-dependent weight matrices: each weight matrix has to implement the transformation of the code correctly, independent of the animal's current position:

$$\boldsymbol{W}(\Delta\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x} + \Delta\boldsymbol{x}) \quad \forall \boldsymbol{x} \tag{4.4}$$

This constraint ensures that if the agent is at a position $\boldsymbol{x}$, it can use $\boldsymbol{W}(\Delta\boldsymbol{x})$ to predict where it will reach next, permitting path-integration. Further, it can be mathematically derived that this constraint forces the code to contain a small number of fourier features, providing a basis for further analysis.

The combination of efficient coding with path-integration leads to a normative theory that is both tractable, and able to capture the multimodular, axis-aligned grid cell structure. Further, it predicts various aspects of the grid cell code, such as that the angle between successive modules should be small. It does not directly explain the conjunctive grid coding, nor are action dependent weight matrices particularly biologically plausible. Both of these problems are somewhat solved through action gating, a plausible scheme to implement action-dependent weight matrices, chapter 2, as seen in other models (Logiaco, Abbott, and Escola, 2021). Hence, it seems that the phenomena could be included in such a model.

Neural Network Models The most common approach is to train a neural network to path-integrate, then study the internal representation of the network. In its simplest version you provide an RNN with a sequence

of actions, and the RNN has to output the current position. This captures all three aspects of the efficient path-integrating code above: the code must path-integrate, it must distinguish different points so they can be decoded, and it must do so using positive activities (if using ReLU nonlinearities) and low weights (if using regularisation). However, the precise design choices, and the results, have varied considerably.

- Some models provide the action as a standard input to the RNN:

$$\boldsymbol{g}(t+1) = \sigma(\boldsymbol{W}\boldsymbol{g}(t) + \boldsymbol{I}\boldsymbol{a}(t) + \boldsymbol{b}) \tag{4.5}$$

  while others learn a mapping between the action and the weight matrix, as in the normative model above.

- Some networks predict (x, y) position, others gaussian place cells, still others difference-of-guassian place cells which bakes in a preference for a particular frequency, section 4.2.3

- Some networks use a ReLU nonlinearity, enforcing nonnegativity of neural activity, others use tanh.

- Weight or activity is often constrained, either through a regularisor, or through a unit norm constraint.

- Other regularisors might be added, most often the conformal isometry loss, which we saw can also bake in a preference for one frequency, section 4.2.4.

An early pair of results provided suggestive findings that these were good models for grid cells. Cueva and Wei (2018) trained agents on exactly this kind of path-integration task using a standard RNN and found grid like and band like neurons, though these grids were often square rather than hexagonal. One key difference here was the use of tanh rather than ReLU nonlinearity, meaning the activities were both positive and negative, and the readout was (x, y) coordinates rather than a place cell type code. Concurrently, Banino et al. (2018) trained a large reinforcement learning model and showed that a feedforward layer in the network, heavily regularised by dropout, learnt to be somewhat griddy (though there have been some concerns about the similarity of these 'grid cells' to low-pass filtered noise (Sorscher, G. Mel, et al., 2019)).

Since then, the class of models that learn an action-dependent weight matrix have been very successful. These models were first studied in the grid cell setting by Issa and K. Zhang, 2012, who derived some conditions for such a model to work, and showed they could implement a system that looked like grid cells. Then, as part of a larger model of the hippocampal-entorhinal system, J. Whittington et al. (2018) and J. C. Whittington, T. H. Muller, et al. (2020) trained sub-networks to path-integrate, and found hexagonal modules of grid cells, though they baked the modular structure into the network. Another vein of work showed that using conformal isometry losses and a difference of gaussian place cell readout you can learn a single module of hexagonal grid cells (Gao, J. Xie, Wei, et al., 2021; Gao, J. Xie, Zhu, et al., 2019; Yilun Xu et al., 2020). Finally, Schaeffer, Khona, Ma, et al. (2023) showed that training the action-dependent matrices in an otherwise standard ReLU RNN with a unit-norm constraint, an activity loss to reduce network capacity, a conformal loss, and a nonlinear separation loss, led to multiple modules of axis aligned grid cells. These models seem to provide a few overlapping sufficient set conditions for grid cells, though given our theoretical work, (Will Dorrell et al., 2023), I think it is likely that there is a smaller necessary set of conditions out there. Further, since these models do not explicitly capture the way velocity is coded by neurons, this architecture will never capture the conjunctive grid cells. Despite this, they present a ringing endorsement for the idea that optimising for a good, efficient, path-integrating code for position is sufficient for recovering grid-cells, even in the RNN setting.

Path-integrating in more standard RNNs has been a more varied storied. Sorscher, G. Mel, et al. (2019) and Sorscher, G. C. Mel, Ocko, et al. (2023) showed that a path-integrating RNN with standard action inputs trained to predict a set of difference-of-gaussian place cells learns a single axis-aligned module of grid cells[3], later supported by (M. Tang, Barron, and Bogacz, 2024). A similar story was seen in Pettersen et al. (2024), who showed that a metric approach combined with path-integration led to a single module of axis-aligned hexagonal grid cells. Both these approaches highlight a move from efficient coding-only approaches to path-integration: the coding losses alone produce hexagonal grid cells without path-integration, but the axes of these grid cells are not aligned. Additionally asking for path-integration aligns the population of grid cells. However, Pettersen et al. (2024) and V. Schøyen et al. (2023) also find that such networks learn a population of band-like cells, and that these are the neurons that seem to do the work of performing path-integration - the network can basically path-integrate without the grid cells! This is in contrast to action-dependent weight models, in which the grid cells are vital for the path-integration. Finally, D. Xu et al. (2025) show that a standard RNN formulation with a unit-norm, positivity, and conformal constraint is sufficient to generate a single module of grid cells, matching theoretical work (V. S. Schøyen et al., 2025). Why some models seem to produce only grid cells (D. Xu et al., 2025), and others find band cells that actually perform path-integration (Pettersen et al., 2024; V. Schøyen et al., 2023), might come down to the choice of a difference-of-gaussian readout. Overall, these

[3]This result was questioned in various ways Schaeffer, Khona, Bertagnoli, et al. (2023), Schaeffer, Khona, and I. Fiete (2022), Schaeffer, Khona, Koyejo, et al. (2023), and Sorscher, G. C. Mel, Nayebi, et al. (2022), but this part seems robust.

models have an advantage over the action-dependent recurrent matrix formulations, as they are able to make predictions about the mechanism for velocity updates. In particular, they show evidence that the velocity update mechanism, using conjunctive heading-grid cells and shifted connectivity, is observed in task-optimised networks too (Sorscher, G. C. Mel, Ocko, et al., 2023). However, none of these models have yet produced a robust multi-modular path-integrating grid cell code, and the discrepancies between models are confusing. Future work will likely solve these problems.

Overall, it seems well established that RNNs optimised to perform a task that includes (1) path-integration, (2) separation of points, and (3) biological constraints, mainly nonnegativity of firing rates and a regularisor on firing, robustly learn grid cells, even multiple modules of them. However, as yet the precise structure of the set of necessary constraints is unclear, especially when using a more standard RNN architecture.

4.4 Discussion

Following over a decade of theoretical and computational work, it seems that normative arguments point to a core claim: grid cells form a (1) high-fidelity, (2) path-integrating, (3) biological code for space. In contrast, normative attempts to explain grid cells without path-integration struggle to capture all aspects of the code. This coheres with mechanistic and perturbative work to support a compelling narrative regarding the grid cell code. Models based on action dependent weight matrices recover the multi-modular axis-aligned structure of grid cells both theoretically (Will Dorrell et al., 2023) and numerically (Schaeffer, Khona, Ma, et al., 2023), but are unable to model the conjunctive grid cells. Models using standard RNNs can fit conjunctive grid cells (Sorscher, G. C. Mel, Ocko, et al., 2023), but are at times badly behaved (Pettersen et al., 2024; Schaeffer, Khona, and I. Fiete, 2022; V. Schøyen et al., 2023), and are yet to generate multiple modules (D. Xu et al., 2025). It seems highly likely that a careful combination of the best parts of both will form a cohesive model for the set of 4 grid cell phenomena we began with.

How constrained are these ideas? Across this body of work, the way in which the three ideas: 'high-fidelity', 'path-integrable', or 'biological', have been formalised has varied. This is a good thing, demonstrating robustness. However, some recurring motifs stand-out. In all cases, the biological constraints limit the capacity of the system (e.g. by limiting the range of firing rates), and ensure the problem is not rotationally invariant, using a nonnegativity constraint either on neural firing or on weights. Similarly, path-integration always implies some mechanism for forward modelling: predicting the next encoding from your previous encoding and an action. Finally, the implementation of a high-fidelity code has been interesting. In all cases that derive multiple modules meaningfully, some form of nonlinearity is necessary (Will Dorrell et al., 2023). A linear decoder (including a population vector decoder), or a linear similarity measure such as the dot product similarity, is not able to exploit the rich positional structure in a multi-modular grid cell code. This ensures that optimising a code subject to a linear readout constraint usually limits the population to a single module. However, when similarity is compared nonlinearly, multiple modules are preferred.

Single Neurons are Pleasingly Constraining The claim above is a great example of using measured neural representations to infer biological mechanisms: since we measure grid cells, there must be a nonlinear position decoding mechanism in the brain, likely the mapping through the dentate gyrus to the hippocampus. The existence of this kind of inference from a set of single cell measurements is exiting. Broadly, it is potentially unclear how much measuring a small number of single neurons can reliably constrain our models. Reviewing this literature we see that it has been incredibly constraining. Fitting just four high-level properties of the system has identified a core set of principles across models, and has proved adept at discounting alternative hypotheses. This is a ringing endorsement for the plodding progress of standard neuroscience.

RNNs as neural models This also speaks to the inclusion of task-optimised neural networks in our plodding progress. Using task-optimised neural networks as models is somewhat controversial: in complex tasks they are often as confusing as the brain, limiting the insights we can gain from them. Yet the grid cell literature presents a compelling case for their power when coupled with clear experimentation, and thorough analysis. Task-optimised networks permit you to try a variety of hypotheses relatively quickly. Their downside is that the signal you measure might have been caused by any number of choices made in architecture, training, or regularisation, and it is often hard to test for all of these. Simplifying the model to the point where theoretical work can be done often provides insight into the core components necessary, allowing fine-tuning of the RNN experiments. Iterations of this cycle seem to have nearly settled on a cohesive framework in the case of grid cells. We are optimists, and hope this will be broadly true. Yet, we note that in the grid cell world this has already taken a decade of intense arguments: it is not necessarily easy.

Place vs. Grid Cells This literature makes some definitive answers to the perenniel questions about the differing roles of grid vs. place cells. In normative theories place cells emerge often, including for neuron-rich efficient coding problems without a baked in frequency prior, section 4.2. Reducing the number of neurons (or equivalently, increasing the resolution of space required from the model) when coupled with a nonlinear decoding mechanism, transitions the optimal efficient code from a place cell code, to a multifield code, as is observed in hippocampus, section 4.2, fig. 4.4. Models with path-integration trace a parallel story. An actionability constraint has no effect on the population if there are infinitely many neurons, and so the solutions are the same as the efficient coding approach: place cells, regardless of the linearity of the readout. However, reducing the neuron number forces the optimal representation to change, the combination of path-integration and a nonlinear decoder leads to multiple modules of grid cells. If the decoder is insufficiently powerful (i.e. linear) instead we are left with a single module of grid cells, fig. 4.4. This suggests a normative model in which hippocampus is forming an efficient code, and grid cells a path-integrable map, matching the many computational models of these systems (J. C. Whittington, T. H. Muller, et al., 2020). In sum: place cells are better than grid cells if you have enough neurons to tile the space to the required resolution, otherwise nonlinear decoders prefer either a multifield place cell, or multiple modules of grid cells if you have to path-integrate, while linear decoders can only use place cells.

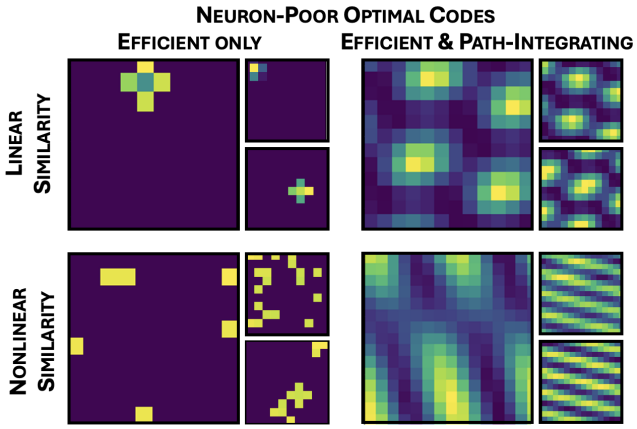


Figure 4.4: We optimise a nonnegative encoding of place to ensure that the similarity of the representation of two points in neural space is sharp and decays for points further apart. We either use the dot-product similarity ($\boldsymbol{z}(\boldsymbol{x})^T\boldsymbol{z}(\boldsymbol{x}')$ - linear) or the exponential of the neural distance (($e^{-\frac{||\boldsymbol{z}(\boldsymbol{x})-\boldsymbol{z}(\boldsymbol{x}')||^2}{2\sigma^2}}$ - nonlinear). When neuron-rich the actionability constraint is automatically satisfied and all optimal codes are place cells, fig. 4.2. When neuron-poor the optimal code varies. Linearity with path-integration leads to a single module of grid cells (which might be a place cell module, since place cells are the lowest frequency grid), while without path-integration we get place cells. With a nonlinear decoder an actionable code forms multiple modules of grids, while an efficient-only code forms random multifield patterns.

This example illustrate part of the power of normative modelling. Previous discussions of place and grid cells have often focused on comparing the two coding schemes, leading to the conclusion that grid cells are the natural efficient code for space (Mathis, Herz, and M. Stemmler, 2012). Normative modelling can search amongst all possible codes, showing that if the only goal was efficiency, the best choice would in fact never be grid cells. Normative modelling forces you to consider many more possibilities.

4.4.1 Future Work

We have focused on models capable of capturing four key phenomena regarding grid cells: their (1) multiple (2) axis-aligned (3) hexagonal modules each with a (4) conjunctive heading-grid partner. This set has proven productive, and versions of these models are able to capture many additional behaviours of grid cells, such as their changing lengthscales in different rooms (Will Dorrell et al., 2023; H. Stensola et al., 2012), the relative lattice lengthscales (Barry et al., 2012; M. Stemmler, Mathis, and Herz, 2015; H. Stensola et al., 2012; Wei, Prentice, and Balasubramanian, 2015) and angles (Will Dorrell et al., 2023), the relationship between number of neurons in a module and field-to-lattice length ratio (Will Dorrell et al., 2023), the relationship between 1D and 2D coding in the same grid module (Rebecca et al., 2025), and the alignment of grid axes to a square room (Rebecca et al., 2025; H. Stensola et al., 2012). Despite this, there remain many grid cell behaviours that are beyond these models, a few of which we highlight now.

Grid Cells in Other Spaces We have focused on grid cells in 2D, a natural question is how might they behave in other spaces. Normative theories of path-integrable representations naturally generalise to other spaces, and almost always predict multiple modules densely packed lattices in other spaces (Will Dorrell et al., 2023; M. Stemmler, Mathis, and Herz, 2015), these match equivalent formulations for the optimal heading direction code (Aceituno, Dall'Osto, and Pisokas, 2024) - an analogous problem for a periodic one-dimensional variable. However, it appears that grid cells are a bespoke 2-dimensional system, it appears that a 1-dimensional map is understood by mapping it onto an optimal 1D slice of the grid lattice choice (Jacob et al., 2019; Rebecca et al., 2025; Yoon et al., 2016). Conversely, 3D grid cells appear to have multiple randomly scattered fields (Ginosar et al., 2021; Grieves et al., 2021), in contrast to either the models discussed so far, and more bespoke projection models (Klukas, Lewis, and I. Fiete, 2020). Models have been proposed that cohesively capture some aspects of both 2D and 3D coding (Ginosar et al., 2021), but as reviewed earlier, they do a poor job at fitting the structured 2D behaviour. Whether there is some preserved structure in the 3D recordings, or a more general model that explains how grid cells encode spaces beyond 2D remains a topic for further work.

Warping of Grid Cells to Environments or Rewards One finding is that grid cells don't always look so... griddy. In trapezoidal environments the lattice bends along the walls (Krupic et al., 2015), the lattice lengthscale gets smaller near boundaries (Hägglund et al., 2019), in large environments there are often inhomogeneities (Gutiérrez-Guzmán, Hernández-Pérez, and Dannenberg, 2025; T. Stensola et al., 2015), though these sometimes disappear with experience (Carpenter et al., 2015), grid fields warp in response to rewards (Boccara et al., 2019), and the metric stretches in inhomogenous environments (J. H. Wen et al., 2024). Some models have taken this at face value, and attempted to normatively explain the warped grid responses, for example as the optimal code for uncertainty (Y. H. Kang, Wolpert, and Lengyel, 2023). Others have argued that the warping is the effect of encoding other variables simulataneously in nearby neural populations, which will optimally mix in some settings (William Dorrell, K. Hsu, et al., 2025; J. C. Whittington, Will Dorrell, et al., 2023). A final class model these effects as a re-centering of the grid code in response to an external cue, such as a boundary (Ocko et al., 2018). Existing grid cell models are consistent with the last two, but not the first type of model: the measured rate maps could just represent the pure grid code, or the grids after a spatially dependent recentering operation, making perfect grids appear bent in some environments or towards some rewards. However, as yet no model is able to bridge these two domains clearly, leaving open the possibility that it is not possible and that our basic models are wrong.

Oscillations The grid code oscillates strongly at the theta frequency (Hafting et al., 2005). Oscillatory interference models foreground this behaviour (Burgess, 2008; Burgess, Barry, and O'keefe, 2007), while continuous attractor models, and correspondingly most normative theories, are consistent with such oscillations but do not rely on it. Vollan et al. (2025) found that the grid code in running rodents alternated on each theta cycle between mapping the rodent's future position to the left and right. It seems that future work will usefully study the structure and function of these oscillating sweeps.

4.4.2 Conclusion

In conclusion, the manifold structures present in the grid cell system have provided impressive constraints for normative theorising. After much work, the field has settled on a consistent set of normative theories: grid cells are a high-fidelity, path-integrable, biological (i.e. constrained and axis-dependent) code for space, agreeing with mechanistic and experimental work. In the future we hope these insights will generalise to grid cells in more complex settings, other neural systems, and provide broad lessons for successful normative theorising.

Part II

Modularity & Mixed Selectivity

The previous chapter was a good example of this thesis' goal: attempting to construct tools for reasoning about the optimal neural implementations of computations. These tools allowed us to turn high level algorithmic ideas, like path-integration, into testable predictions about single neuron coding.

A key assumption in the previous chapter, shared with most neuroscience, is modularity: the idea that you can break the brain's function into meaningful subparts that can be understood in isolation[4]. The previous section only worked because we assumed the existence of an isolated set of neurons that performed path-integration. A more complete theory would prescribe not only the implementation of each these subparts, but what the meaningful subparts are. In this chapter we take partial steps in this direction.

Whether (or perhaps more precisely: the extent to which) neurons meaningfully decompose into disjoint populations performing meaningful computations is currently much debated. On the one hand our brains are clearly modular. At the macroscale, different regions, such as visual or language cortex, perform specialised roles; at the microscale, single neurons often precisely encode single variables such as self-position (Hafting et al., 2005) or the orientation of a visual edge (Hubel and Wiesel, 1962). This mysterious alignment between meaningful concepts and single neuron activity has for decades fuelled hope for understanding a neuron's function by finding its associated concept. Yet, as neural recording technology has improved, it has become clear that many neurons behave in ways that elude such simple categorisation: they appear to be mixed selective, responding to a mixture of variables in linear and nonlinear ways (Rigotti et al., 2013; Tye et al., 2024). Most puzzlingly, grid cells, whose behaviour is the epitome of a modular neocortical computation, seem to switch into mixed-selectivity when rewards are distributed inhomogeneously in the environment (Boccara et al., 2019; Butler, Hardcastle, and Giocomo, 2019).

The same modules vs. mixtures debate has recently been reprised in two separate subfields of machine learning. The first, Mechanistic Interpretability, seeks to understand artificial neural networks like a neuroscientist would a brain. There, employing similar 'neural tuning curve' approaches has led to similarly varied findings, with some studies showing meaningful single unit responses (Bau et al., 2020; Olah, Mordvintsev, and Schubert, 2017) and others not (Dey et al., 2025). The second community, Disentangled Representation Learning, approaches the issue from the other direction. Rather, than trying to understand why a measured representation might or might not be modular, they try to design models that will decompose data into its meaningful subparts. The pertinent question in this field is then 'how should I process my data to extract the modular structure'. From this perspective it is clear that clarity on what makes a representation modular, as studied in neuroscience or mechanistic interpretability, can be reused to create disentangling tools. In a further point of synergy between these subfields, recent work has sought to understand puzzling mixed-selective representations in neural networks by disentangling them into meaningful subparts (Bricken et al., 2023).

This leads us to our research question for this chapter: why are neurons, biological or artificial, sometimes modular and sometimes mixed selective? Here, we develop a theory that, under particular conditions, precisely determines, for any dataset, whether the optimal learned representations will be modular or not. More precisely, in the linear autoencoder setting, we show that modularity in biologically constrained representations is governed by a *sufficient spread* condition that can roughly be thought of as measuring the extent to which the *range* of the source variables (sources, aka factors of variation) is *rectangular*. We put this theory to work understanding a variety of pieces of neural data and we end with a novel identifiability criterion for related matrix factorisation problems

We iteratively develop our results over the following sections:

- In chapter 5 we establish our basic theoretical results on modularisation in biological linear autoencoders.
- Then in chapter 6 we extend our theoretical results to linear recurrent settings, as considered in much of the rest of this thesis. This allows us to understand the modularisation of grid cells from the previous chapter, and will form the basis for the modularisation of prefrontal memory codes in the following chapter.
- In chapter 7 we dive into neuroscience. We frame these findings as a theory of functional cell types, including providing a potential explanation for why grid cells might sometimes be modular and other times mixed selective to position and reward.
- Finally in chapter 8 we move back to maths and show that a large set of neural optimisation problems, including the linear autoencoder studied here, can be framed as convex optimisations. We use this to derive more general necessary and sufficient conditions for modularisation (in the context of linearly mixed sources). This is of independent interest for matrix factorisation as the first necessary and sufficient characterisation of model identifiability. Finally, we use this result to derive an identifiability result on neural tuning curves: i.e. in this setting we answer the question 'what must be true about a set of tuning curves such that every optimal solution will contain a neuron tuned like this.

[4]It is unclear to me if this is a statement about the world (i.e. to be understandable a complicated system must be composed of simpler subparts that can be combined to understand the whole) or a statement about the way in which we understand (i.e. if a human has understood a complicated system it was because it was able to design good abstractions into which to break the system).

- We end in chapter 9 with a broader discussion of related work and implications, in particular regarding theories of mixed selectivity in the brain.

The analysis in this chapter (like this thesis) relies on some form of linearity in the problem, a feature often not shared with reality. In a series of published not presented in this thesis my collaborators and I have shown that these results somehwat generalise to nonlinear settings (William Dorrell, K. Hsu, et al., 2025; J. C. Whittington, Will Dorrell, et al., 2023). Further, we turned these insights into a state-of-the-art disentangling method (K. Hsu, William Dorrell, et al., 2023). We return to these in the discussion, but will not discuss them in detail.

In sum, this work aims to show how the same efficient computing approaches developed throughout this thesis can be used in certain settings to describe not only how a single meaningful module will be implemented, but to reason about what we should consider to be the neurally meaningful modules.

4.5 A Note on Different Notions of Independence

We will use multiple different notions of independence between variables in this work, each of which can be defined analogously to statistical independence, fig. 4.5:

- Two variables are statistically independent if no matter what value one variable takes, the distribution of the other is the same: $p(s_1, s_2) = p(s_1)p(s_2)$, therefore $p(s_1|s_2 = x) = p(s_1|s_2 = y) = p(s_1)$.
- Two variables are support independent if no matter what value one variable takes, the support of the other is the same.
- Two variables are range independent if no matter what value one variable takes, the range of the other is the same.
- Two variables are extreme point independent if across datapoints when one variable takes its maximum or minumum values, the other also takes its maximum of minimum values. In other words, the joint distribution has support at the four extremal corners.

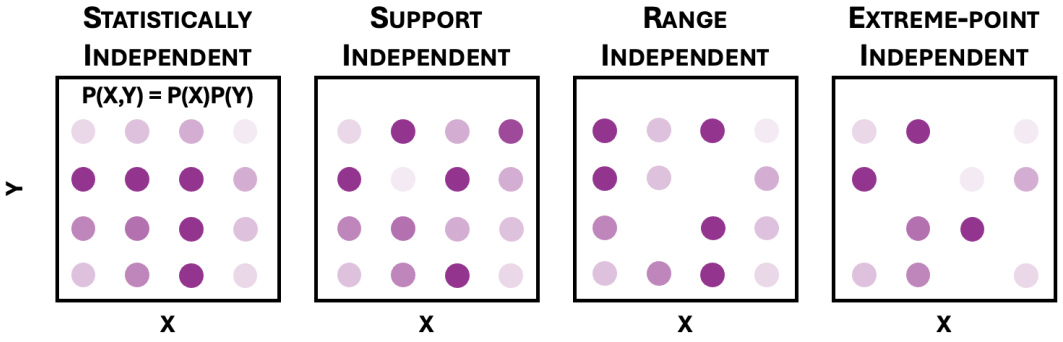


Figure 4.5: Statistical independence implies the probability factorises. Support independence implies that the support of one variable doesn't change depending on the other, though the probabilities might (in contrast to statistical independence). Range independence means the max and min value of one variable is independent of the other. Finally, extreme-point independence means only that the four corners have non-zero probability: the range of one variable at the other variable's max and min are its own max and min.

Chapter 5

Sufficient Spread implies Modularity

Why do biological and artificial neurons sometimes modularise, each encoding a single meaningful variable, and sometimes entangle their representation of many variables? Here, we present one answer to this question. We develop a theory of when biologically inspired networks—those that are nonnegative and energy efficient—modularise their representation of source variables (sources). We derive necessary and sufficient conditions on a sample of sources that determine whether the neurons in an optimal biologically-inspired linear autoencoder modularise. Our theory applies to any dataset, showing that sources modularise if their support is "sufficiently spread". Some simpler sufficient conditions that satisfy this spread condition include support independent variables (which includes statistically independent variables). These theories are verified empirically, and intuition is presented.

5.1 Modularisation in Biological Linear Autoencoders

We begin with our main technical result: necessary and sufficient conditions for the modularisation of biologically inspired linear autoencoders.

5.1.1 Preliminaries

Let $\boldsymbol{s} \in \mathbb{R}^{d_s}$ be a vector of d_s scalar source variables (sources, aka factors of variation). We are interested in how the empirical distribution, $p(\boldsymbol{s})$, affects whether the sources' neural encoding (latents), $\boldsymbol{z} \in \mathbb{R}^{d_z}$, are *modular* with respect to the sources, i.e., whether each neuron (latent) is functionally dependent on at most one source. We build a simplified model in which neural firing rates perfectly linearly autoencode the sources while maintaining nonnegativity (since firing rates are nonnegative):

$$\boldsymbol{z} = \boldsymbol{W}_{\text{in}}\boldsymbol{s} + \boldsymbol{b}_{\text{in}}, \quad \boldsymbol{s} = \boldsymbol{W}_{\text{out}}\boldsymbol{z} + \boldsymbol{b}_{\text{out}}, \quad \boldsymbol{z} \geq 0. \tag{5.1}$$

Subject to the above constraints, we study the representation that uses the least energy, as in the classic efficient coding hypothesis (Barlow et al., 1961). We quantify this using the ℓ^2 norm of the firing rates and weights (other activity norms considered later):

$$\min_{\boldsymbol{W}_{\text{in}},\boldsymbol{b}_{\text{in}},\boldsymbol{W}_{\text{out}},\boldsymbol{b}_{\text{out}}} \left\langle ||\boldsymbol{z}||_2^2 \right\rangle_{p(\boldsymbol{s})} + \lambda \left(||\boldsymbol{W}_{\text{out}}||_F^2 + ||\boldsymbol{W}_{\text{in}}||_F^2 \right) \text{ s.t. eq. (5.1).} \tag{5.2}$$

We remark that there are common analogues of representational nonnegativity and weight energy efficiency in modern machine learning (ReLU activation functions and weight decay, respectively).

5.1.2 Intuition I: A Visual Argument for Modularisation of Independent Sources

We now present an intuitive argument that in this setting independent variables should modularise. The intuition follows a visual proof, fig. 5.1, whose key ideas can be seen with just two neurons encoding two sources. To autoencode perfectly the encoding must represent both sources, as in fig. 5.1 left. To satisfy the nonnegativity constraint the encoding has to be shifted into the positive quadrant, fig. 5.1 centre. In particular, the bias added to achieve this should be the minimal bias that makes each z_i nonnegative for all values of s_1 and s_2, since any additional bias will increase the firing costs for no gain. This corresponds to the grey square touching each axis, rather than floating off into the positive orthant. Since the two sources are independent, a mixed neuron in the representation implies a large bias and hence a large activity cost, as seen in the figure by the cost of misaligning the corners of the square with the neuron axes. It is therefore lower cost, as in fig. 5.1 right, to align the axes of the sources with those of the neurons, so that each bias is only offsetting a single source. This argument

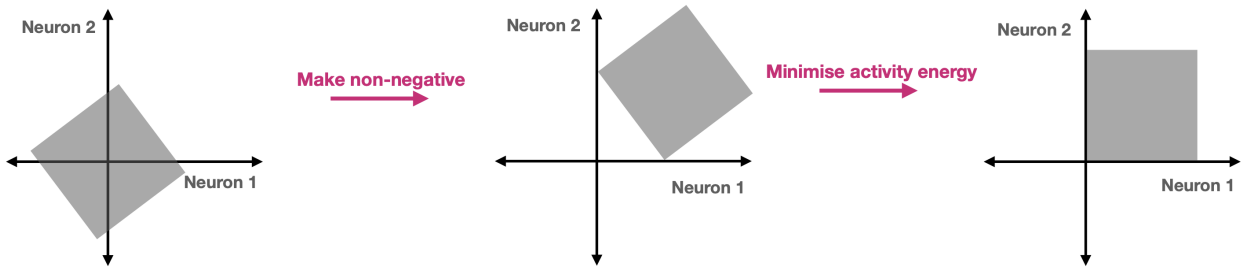


Figure 5.1: **Proof intuition.** Two uniformly distributed independent factors represented with two entangled neurons (left). The representation can be made nonnegative at the expense of activity energy (middle). Activity energy is minimised under a nonnegativity constraint when the neurons are axis aligned to task factors (i.e. disentangled, right). Grey boxes denote uniform distributions over neural activity induced by uniform distributions over task factors. Note our proof does not require uniformity.

is similar to Pehlevan, Mohan, and Chklovskii (2017) in nonnegative ICA, we review the related literature in chapter 9. Further, we note that these arguments also work if, rather than strict nonnegativity, the neural firing rates fluctuate around a baseline. In this setting the neural activity could be negative, corresponding to fluctuation below the baseline, but would still have a constraint not to become too negative, permitting the use of this argument.

5.1.3 Intuition II: Source Support Governs Modularisation

This argument provides some intuition about why independent variables modularise. Further, it makes clear the key role of the support, and in particular the notion of support independence. If two variables are support independent their support forms a square, and it seems reasonable to think the same argument as in fig. 5.1 applies. But to get more precise necessary and sufficient conditions we'll have to hone our intuition further. Consider a hypothetical mixed selective neuron,

$$z_j = w_{j1}s_1 + w_{j2}s_2 + b_j, \tag{5.3}$$

It is functionally dependent on both s_1 and s_2, i.e. it is mixed selective. Perhaps, however, a modular representation, in which this neuron is broken into two separate modular encodings, would be better. We can create such a solution with two neurons each coding a single source:

$$z_{j'} = w_{j1}s_1 + b_{j'}, \quad z_{j''} = w_{j2}s_2 + b_{j''}. \tag{5.4}$$

For simplicity, we assume the two sources are linearly uncorrelated, have mean zero, and are supported on an interval centered at zero (fig. 5.2a; we relax these assumptions in our full theory). Then, for fixed w_{j1} and w_{j2}, the optimal (energy efficient) bias should be large enough to keep the representation nonnegative, but no larger:

$$b_j = -\min_{s_1,s_2}\left[w_{j1}s_1 + w_{j2}s_2\right], \quad b_{j'} = -\min_{s_1} w_{j1}s_1, \quad b_{j''} = -\min_{s_2} w_{j2}s_2, \tag{5.5}$$

where the minimisations are over the empirical distribution $p(s_1, s_2)$. Now that both representations are specified, we can compare their costs. In our problem modularity is driven only by the activity loss (section 5.A justifies this claim). Further, in this simplified setting, most terms in the activity loss are zero or cancel, and one finds that the mixed selective eq. (5.3) case uses less energy than the modular eq. (5.4) when

$$b_j^2 < b_{j'}^2 + b_{j''}^2. \tag{5.6}$$

The key takeaway lies in how b_j is determined by a joint minimization over s_1 and s_2 eq. (5.5). Assume positive w_{j1} and w_{j2}; then if s_1 and s_2 take their minima simultaneously, as in fig. 5.2c, the mixed bias must be large:

$$b_j = -\min_{s_1,s_2}\left[w_{j1}s_1 + w_{j2}s_2\right] = -\min_{s_1}[w_{j1}s_1] - \min_{s_2}[w_{j2}s_2] = b_{j'} + b_{j''} \tag{5.7}$$

In this case, the energy of the mixed solution will always be worse than the modular, since $b_j^2 = (b_{j'} + b_{j''})^2 > b_{j'}^2 + b_{j''}^2$. Alternatively, mixing will be preferred when s_1 and s_2 *do not* take on their most negative values at the same time, as in fig. 5.2d, since then b_j can be smaller while maintaining positivity, and the corresponding energy saving satisfies the key inequality eq. (5.6).

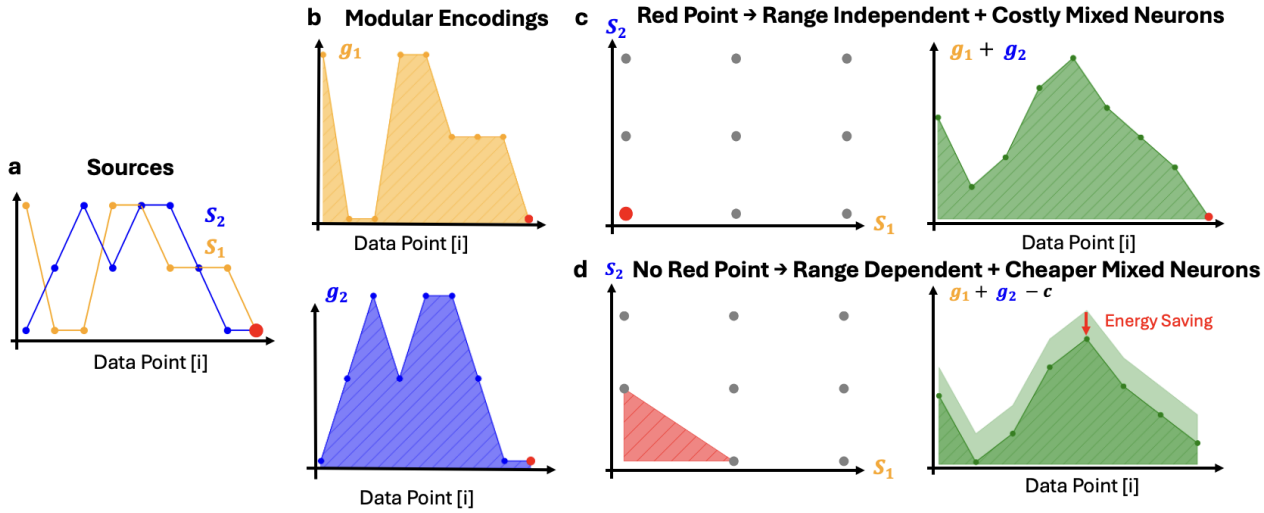


Figure 5.2: Left: Values of two sources across a dataset and their modular encoding with associated costs (shaded regions). Top Right: If the dataset includes the red datapoint (left) then the two variables take their minimal value at the same time, and the mixed encoding must include a large bias, using more energy (right). Bottom Right: If the red point is instead missing (left) then the mixed encoding can use a smaller bias while remaining positive and save energy (right).

5.1.4 Precise Conditions for Modularising Biological Linear Autoencoders

We now make precise the intuition developed above. Proof is deferred until section 5.A.

Theorem 1. *Let* $\boldsymbol{s} \in \mathbb{R}^{d_s}$, $\boldsymbol{z} \in \mathbb{R}^{d_z}$, $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{d_z \times d_s}$, $\boldsymbol{b}_{\text{in}} \in \mathbb{R}^{d_z}$, $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{d_s \times d_z}$, *and* $\boldsymbol{b}_{\text{out}} \in \mathbb{R}^{d_s}$, *with* $d_z > d_s$. *Consider the constrained optimization problem*

$$\begin{aligned} \min_{\boldsymbol{W}_{\text{in}}, \boldsymbol{b}_{\text{in}}, \boldsymbol{W}_{\text{out}}, \boldsymbol{b}_{\text{out}}} \quad & \left\langle ||\boldsymbol{z}^{[i]}||_2^2 \right\rangle_i + \lambda \left(||\boldsymbol{W}_{\text{in}}||_F^2 + ||\boldsymbol{W}_{\text{out}}||_F^2 \right) \\ \text{s.t.} \quad & \boldsymbol{z}^{[i]} = \boldsymbol{W}_{\text{in}} \boldsymbol{s}^{[i]} + \boldsymbol{b}_{\text{in}}, \; \boldsymbol{s}^{[i]} = \boldsymbol{W}_{\text{out}} \boldsymbol{z}^{[i]} + \boldsymbol{b}_{\text{out}}, \; \boldsymbol{z}^{[i]} \geq 0, \end{aligned} \tag{5.8}$$

where i *indexes a finite set of samples of* $\boldsymbol{s}$. *At the minima of this problem, the representation modularises, i.e. each row of* $\boldsymbol{W}_{\text{in}}$ *has at most one non-zero entry, iff the following inequality is satisfied for all* $\boldsymbol{w} \in \mathbb{R}^{d_s}$*:*

$$\left(\min_i [\boldsymbol{w}^\top \bar{\boldsymbol{s}}^{[i]}] \right)^2 + \sum_{j,j' \neq j}^{d_s} w_j w_{j'} \left\langle \bar{s}_j^{[i]} \bar{s}_{j'}^{[i]} \right\rangle_i \geq \sum_{j=1}^{d_s} \left(w_j \min_i \bar{s}_j^{[i]} \right)^2, \tag{5.9}$$

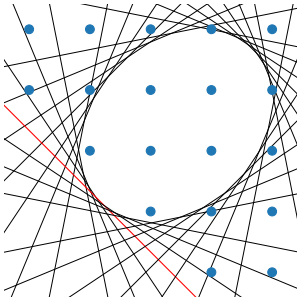
with equality only for $\boldsymbol{w} \propto \boldsymbol{e}_k$, *defining* $\bar{\boldsymbol{s}} := \boldsymbol{s} - \left\langle \boldsymbol{s}^{[i]} \right\rangle_i$ *and assuming w.l.o.g. that* $\left| \min_i \bar{s}_j^{[i]} \right| \leq \max_i \bar{s}_j^{[i]} \; \forall j \in [d_s]$.

Our theory prescribes a set of inequalities eq. (5.9) that determine whether an optimal representation is modular. These inequalities come from the difference in activity energy between a modular and mixed solutions, just like the intuition we built up in Section section 5.1.3, and in particular eq. (5.6). If a single inequality is broken, the optimal representation is mixed (at least in part); else, it is modular: optimally, each neuron's activity is a function of a single source. These inequalities depend on two key properties of the sources: the shape of the source distribution's support in extremal regions, and the pairwise source correlations. Remarkably, they do not depend on the energy tradeoff parameter λ. These inequalities can be visualised, as done in the wrapped figure below. For a given dataset, $\{\boldsymbol{s}^{[i]}\}$ (blue dots), we can calculate $\left\langle \bar{s}_j^{[i]} \bar{s}_{j'}^{[i]} \right\rangle_i$ and each $\min_i \bar{s}_j^{[i]}$ as they are simple functions of the dataset. For all unit $\boldsymbol{w}$, we can draw the line

$$\boldsymbol{w}^\top \boldsymbol{x} = \sqrt{\sum_{j=1}^{d_s} \left(w_j \min_i \bar{s}_j^{[i]} \right)^2 - \sum_{j,j' \neq j}^{d_s} w_j w_{j'} \left\langle \bar{s}_j^{[i]} \bar{s}_{j'}^{[i]} \right\rangle_i} = \sqrt{\boldsymbol{w}^T \boldsymbol{F} \boldsymbol{w}}, \tag{5.10}$$

where we have defined the following matrix:

$$\boldsymbol{F} = \begin{bmatrix} (\min_i s_1^{[i]})^2 & -\langle s_1^{[i]} s_2^{[i]} \rangle_i & -\langle s_1^{[i]} s_3^{[i]} \rangle_i & \dots \\ -\langle s_1^{[i]} s_2^{[i]} \rangle_i & (\min_i s_2^{[i]})^2 & -\langle s_2^{[i]} s_3^{[i]} \rangle_i & \dots \\ -\langle s_1^{[i]} s_3^{[i]} \rangle_i & -\langle s_2^{[i]} s_3^{[i]} \rangle_i & (\min_i s_3^{[i]})^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5.11}$$



Each $\boldsymbol{w}$ gives us a line, and if there is at least one line that bounds the source support (such as the red one), then an inequality is broken and the optimal representation is mixed. Else it will be modular. This exercise also motivates the following equivalent statement of our conditions.

Theorem 2. *In the same setting as theorem 1, define the set* $E = \{\boldsymbol{y} : \boldsymbol{y}^T \boldsymbol{F}^{-1} \boldsymbol{y} = 1\}$. *Then an equivalent statement of theorem 1 is the representation modularises iff* E *lies within the convex hull of the datapoints, with intersection only permitted at points where the tangent to* E *is orthogonal to a basis direction.*

This therefore provides a simple test: create $\boldsymbol{F}$ and draw the set E which, if $\boldsymbol{F}$ is positive definite (as it often is), is an ellipse as shown in the wrapped figure above. The optimal representation modularises iff the convex hull of the datapoints encloses E, fig. 5.3a. If $\boldsymbol{F}$ has some positive and some negative eigenvalues, then the set E is unbounded, and the optimal representation must mix.

5.1.5 Range Independent Variables Modularise

To clarify our result we present a particularly clean special case.

Corollary 2.1. *In the same setting as theorem 1 the optimal representation modularises if all sources are pairwise extreme-point independent, i.e. if for all* $j, j' \in [d_s]^2$*:*

$$\min_i \left[s_j^{[i]} \middle| s_{j'}^{[i]} \in \left\{ \max_{i'} s_{j'}^{[i']}, \min_{i'} s_{j'}^{[i']} \right\} \right] = \min_i s_j^{[i]}. \tag{5.12}$$

In other words, if the joint distribution is supported on all extremal corners, the optimal representation modularises.

As presented, our theory has some limitations. It focuses on sources that are 1-dimensional, and uses a specific choice of activity norm, the L2, when others might be more reasonable. In fact, however, the core result that it is the independence of the variables' range (i.e. how rectangular they are) that drives modularity is very broadly true. In section 5.A.4 we show that this core result generalises to other activity norms, and in section 5.A.5 we show that it also applies to the modularisation of multi-dimensional variables such as angles on a 2D circle. Thus, our key takeaway is that the independence of variables' range is what determines whether the optimal representation is modular.

5.1.6 Validation of linear autoencoder theory

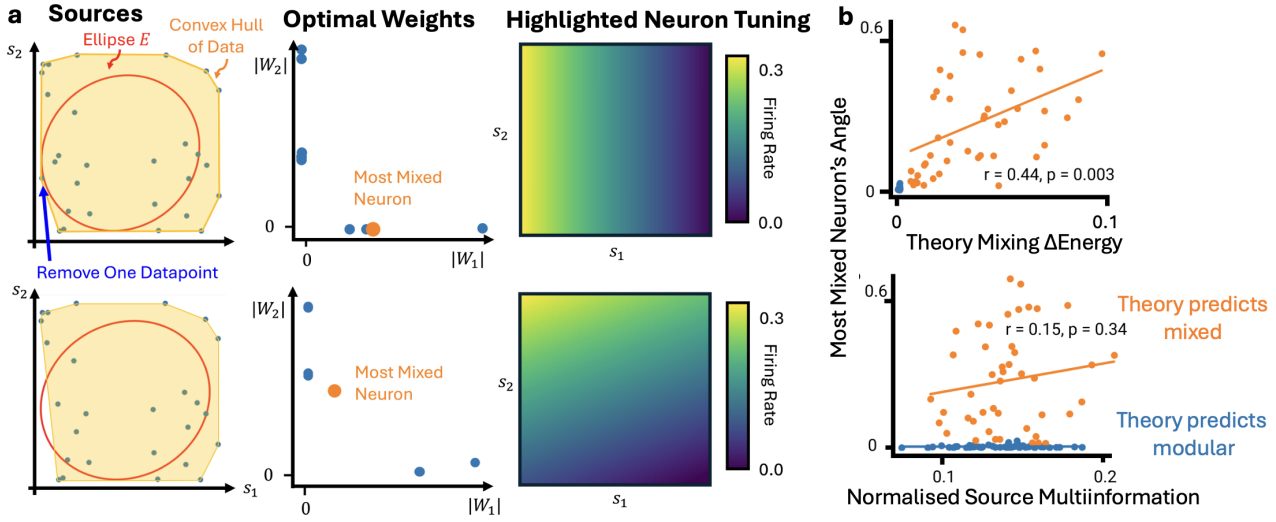


Figure 5.3: Left: Values of two sources across a dataset and their modular encoding with associated costs (shaded regions). Top Right: If the dataset includes the red datapoint (left) then the two variables take their minimal value at the same time, and the mixed encoding must include a large bias, using more energy (right). Bottom Right: If the red point is instead missing (left) then the mixed encoding can use a smaller bias while remaining positive and save energy (right).

We show our inequalities correctly predict modularisation. In particular, as an illustration of the precision of our theory, we create a dataset which transitions from inducing modularising to mixing via the removal of a single critical datapoint (fig. 5.3a). Further, we generate many datasets, create the optimal representation,

and measure the angle θ between the most-mixed neuron's weight vector and its closest source direction, a proxy for modularity. The left of fig. 5.3b shows that our theory correctly predicts which datasets are modular ($\theta = 0$). Further, despite our theory being binary (will it modularise or not?), empirically we see that the degree to which the inequalities in theorem 1 are broken is a good proxy for how mixed the optimal representation is. Finally, on the bottom right of fig. 5.3b we show that on the same datasets a simpler measure of source statistical interdependence does not predict modularisation.

5.A Mathematical Appendices: Positive Linear Autoencoders

In this section we derive our main mathematical results. We prove the main theorem, section 5.A.1, the equivalent statement in terms of the convex hull of the data, section 5.A.2, and the corollary on range-independent variables section 5.A.3. We then generalise this range-independent result to all activity norms, section 5.A.4, and to multidimensional variables, section 5.A.5.

5.A.1 Derivation of Modularisation Conditions

Our proof of the main result, theorem 1, takes the following strategy. First, we show sufficieny. We show that for a fixed encoding size, the weight loss is always minimised by orthogonalising, one example of which is modularising. Then we study when the activity loss is also minimised by modularising, and use that to tell us about the minima of the full loss. Second, we show necessity, we start at the optimal modular representation and show that the same conditions determine whether perturbing will reduce the loss.

Weight Losses are Minimised by Modularising

First, we show a lemma: that, for a fixed encoding size, the weight loss is minimised by orthogonalising the encoding of each source. Writing the representation as an affine function of the de-meaned sources:

$$\boldsymbol{z}^{[i]} = \sum_{j=1}^{d_s} \boldsymbol{u}_j s_j^{[i]} + \boldsymbol{b} = \sum_{j=1}^{d_s} \boldsymbol{u}_j (s_j^{[i]} - \langle s_j^{[k]} \rangle_k) + \boldsymbol{b}' = \sum_{j=1}^{d_s} \boldsymbol{u}_j \bar{s}_j^{[i]} + \boldsymbol{b}' \tag{5.13}$$

then by fixed encoding size we mean that each $|\boldsymbol{u}_j|$ is fixed. We express things in terms of mean-zero variables $\bar{s}_j^{[i]}$ since it is easier and we can freely shift by a constant offset.

Lemma 3 (Modularising Minimises Weight Loss)**.** *The weight loss, $||\boldsymbol{W}_{\text{out}}||_F^2 + ||\boldsymbol{W}_{\text{in}}||_F^2$, is minimised, for fixed encoding magnitudes, $|\boldsymbol{u}_j|$, by modularising the representation.*

Proof. First, we know that $\boldsymbol{W}_{\text{in}}$ must be of the following form:

$$\boldsymbol{W}_{\text{in}} = \begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_{d_s} \end{bmatrix} \tag{5.14}$$

And therefore for a fixed encoding size the input weight loss is constant:

$$||\boldsymbol{W}_{\text{in}}||_F^2 = \text{Tr}[\boldsymbol{W}_{\text{in}}^T \boldsymbol{W}_{\text{in}}] = \sum_{j=1}^{d_s} |\boldsymbol{u}_j|^2 \tag{5.15}$$

So let's study the output weights, these are defined by the way they map the representation back to the sources:

$$\boldsymbol{W}_{\text{out}} \cdot \boldsymbol{z}^{[i]} + \boldsymbol{b}_{\text{out}} = \begin{bmatrix} s_1^{[i]} \\ \vdots \\ s_{d_s}^{[i]} \end{bmatrix} \tag{5.16}$$

The min-norm $\boldsymbol{W}_{\text{out}}$ with this property is the Moore-Penrose Pseudoinverse, i.e. the matrix:

$$\boldsymbol{W}_{\text{out}} = \begin{bmatrix} \boldsymbol{U}_1^T \\ \vdots \\ \boldsymbol{U}_{d_s}^T \end{bmatrix} \tag{5.17}$$

Where each pseudoinverse $\boldsymbol{U}_i$ is defined by $\boldsymbol{U}_i^T \boldsymbol{u}_j = \delta_{ij}$ and lying completely within the span of the encoding vectors $\{\boldsymbol{u}_j\}$. We can calculate the norm of this matrix:

$$|\boldsymbol{W}_{\text{out}}|_F^2 = \text{Tr}[\boldsymbol{W}_{\text{out}} \boldsymbol{W}_{\text{out}}^T] = \sum_{j=1}^{d_s} |\boldsymbol{U}_j|^2 \tag{5.18}$$

Now, each of these capitalised pseudoinverse vectors must have some component along its corresponding lower case vector, and some component orthogonal to that:

$$\boldsymbol{U}_j = \frac{1}{|\boldsymbol{u}_j|^2}\boldsymbol{u}_j + \boldsymbol{u}_{j,\perp} \tag{5.19}$$

We've chosen the size of the component along $\boldsymbol{u}_j$ such that $\boldsymbol{U}_j^T\boldsymbol{u}_j = 1$, and the $\boldsymbol{u}_{j,\perp}$ is chosen so that $\boldsymbol{U}_j^T\boldsymbol{u}_k = \delta_{jk}$. Now, for a fixed size of $|\boldsymbol{u}_j|$, this sets a lower bound on the size of the weight matrix:

$$|\boldsymbol{W}_{\text{out}}|_F^2 = \sum_{j=1}^{d_s}\frac{1}{|\boldsymbol{u}_j|^2} + |\boldsymbol{u}_{j,\perp}|^2 \geq \sum_{j=1}^{d_s}\frac{1}{|\boldsymbol{u}_j|^2} \tag{5.20}$$

And this lower bound is achieved whenever the $\{\boldsymbol{u}_j\}_{j=1}^{d_s}$ vectors are orthogonal to one another, since then $\boldsymbol{u}_{j,\perp} = 0$. Therefore, we see that, for a fixed size of encoding, the weight loss is minimised when the encoding vectors are orthogonal, and that is achieved when the code is modular. □

Activity Loss

Now we will turn to studying the activity loss, and find conditions under which a modular representation is better than a mixed one. We compare two representations, a mixed neuron:

$$z_n^{[i]} = \sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} + \Delta_n = \sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} - \min_i[\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]}] \tag{5.21}$$

And another representation in which we break this neuron apart into its modular form, preserving the encoding size of each source:

$$\boldsymbol{z}_n^{[i]} = \begin{bmatrix} u_{n1}\bar{s}_1^{[i]} \\ \vdots \\ u_{nd_s}\bar{s}_{d_s}^{[i]} \end{bmatrix} - \begin{bmatrix} |u_{n1}|\min_j \bar{s}_1^{[j]} \\ \vdots \\ |u_{nd_s}|\min_j \bar{s}_{d_s}^{[i]} \end{bmatrix} \tag{5.22}$$

The activity loss difference between the modular and mixed representations is:

$$\sum_j u_{nj}^2(\min_i \bar{s}_j^{[i]})^2 - (\min_i[\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]}])^2 - \sum_{j=1,k\neq j}^{d_s} u_{nj}u_{nk}\langle \bar{s}_j^{[i]}\bar{s}_k^{[i]}\rangle_i \tag{5.23}$$

So the modular loss is smaller if, for all $\boldsymbol{u}_n \in \mathbb{R}^{d_s}$:

$$(\min_i[\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]}])^2 > \sum_j u_{nj}^2(\min_i \bar{s}_j^{[i]})^2 - \sum_{j,k\neq j} u_{nj}u_{nk}\langle \bar{s}_j^{[i]}\bar{s}_k^{[i]}\rangle_i \tag{5.24}$$

Further, we can see that if this equation holds for one vector, $\boldsymbol{u}$, it also holds for any scaled version of that vector, therefore we can equivalently require the inequality to be true only for unit norm vectors. This has to hold for all $\boldsymbol{u}$ that are not aligned with the basis vectors, clearly if $\boldsymbol{u} \propto \boldsymbol{e}_k$ the two are equal, since the neuron was already modular.

Sufficient Condition: Combination of Weight and Activity

So, if these inequalities are satisfied you can always decrease the loss by modularising a mixed neuron. Therefore, both the weight and activity loss are minimised at fixed encoding size by modularising, and this holds for all encoding sizes, therefore the optimal solution must be modular. Hence these conditions are sufficient for modular solutions to be optimal.

Necessary Condition

We now prove these conditions are necessary. We show that if one of the inequalities is broken we can create a new neuron in a way that reduces the loss. Imagine rotating the encoding of each source slightly by a vector of angles $\boldsymbol{\theta}$ to create a neuron mixed in just the right way that breaks the inequality, $\boldsymbol{u}$:

$$\boldsymbol{z}^{[i]} = \sum_{j=1}^{d_S} d_j\cos(\theta_j)\boldsymbol{e}_j(s_j^{[i]} - \min_i[s_j^{[i]}]) + \underbrace{\boldsymbol{e}_{d_S+1}\left(\boldsymbol{u}^T\boldsymbol{s}^{[i]} - \min_i[\boldsymbol{u}^T\boldsymbol{s}^{[i]}]\right)}_{\text{new neuron}} \qquad \boldsymbol{u} = \begin{bmatrix} d_1\sin(\theta_1) \\ \vdots \\ d_n\sin(\theta_n) \end{bmatrix} \tag{5.25}$$

Since the inequality is broken we know that the activity loss will decrease at second order in $\boldsymbol{\theta}$. Further, since the size of the encoding is unchanged, so too is $||\boldsymbol{W}_{\text{in}}||_F^2$. Finally, $||\boldsymbol{W}_{\text{out}}||_F^2$ is changed, but only at higher order. To see this consider the pseudoinverse vectors. These will continue to point along the modular neuron directions with a slightly larger length:

$$||\boldsymbol{W}_{\text{out}}||_F^2 = \sum_{j=1}^{d_s} \frac{1}{d_j^2 \cos^2(\theta_j)} = \sum_{j=1}^{d_s} \frac{1}{d_j^2} + \mathcal{O}(\boldsymbol{\theta}^4) \tag{5.26}$$

Hence, to second order in $\boldsymbol{\theta}$, if an inequality is broken we can reduce the loss by moving away from modularity, showing the necessity of these inequalities.

5.A.2 Equivalence of Convex Hull Formulation

In this section we prove theorem 2, an equivalent formulation of the infinitely many inequalities in theorem 1 in terms of the convex hull, restated below:

Theorem 4. *In the setting as theorem 1, define the following matrix:*

$$\boldsymbol{F} = \begin{bmatrix} (\min_i s_1^{[i]})^2 & -\langle s_1^{[i]} s_2^{[i]} \rangle_i & -\langle s_1^{[i]} s_3^{[i]} \rangle_i & \dots \\ -\langle s_1^{[i]} s_2^{[i]} \rangle_i & (\min_i s_2^{[i]})^2 & -\langle s_2^{[i]} s_3^{[i]} \rangle_i & \dots \\ -\langle s_1^{[i]} s_3^{[i]} \rangle_i & -\langle s_2^{[i]} s_3^{[i]} \rangle_i & (\min_i s_3^{[i]})^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5.27}$$

Define the set $E = \{\boldsymbol{y} : \boldsymbol{y}^T \boldsymbol{F}^{-1} \boldsymbol{y} = 1\}$. Then an equivalent statement of the modularisation inequalities eq. (5.9)*, is the representation modularises iff E lies within the convex hull of the datapoints, with intersection only permitted at points where the tangent to E is orthogonal to a basis direction.*

Notice that the matrix $\boldsymbol{F}$ is not guaranteed to be positive definite, meaning E is not necessarily an ellipse. In these cases no convex hull can contain the unbounded quadric it forms, and the optimal solution is mixed regardless of the particular distribution of the datapoints.

We'll begin by rewriting the inequalities, then we'll show each direction of the equivalence.

Rewrite Inequality Constraints

The current form of the inequality is that, $\forall \boldsymbol{w} \in \mathbb{R}^n$:

$$(\min_i \boldsymbol{w}^T \boldsymbol{s}^{[i]})^2 + \sum_{j,j' \neq j}^{d_s} w_j w_{j'} \langle s_j^{[i]} s_{j'}^{[i]} \rangle_i \geq \sum_j w_j^2 (\min_i s_j^{[i]})^2 \tag{5.28}$$

with equality only for $\boldsymbol{w} \propto \boldsymbol{e}_k$. If this equation holds for one vector, $\boldsymbol{w}$, it also holds for all positively scaled versions of the vector, $\boldsymbol{v} = \lambda \boldsymbol{w}$ for $\lambda > 0$, so we can equivalently state that this condition must only hold for unit vectors.

Construct the matrix $\boldsymbol{F}$ as above eq. (5.27), then this condition can be rewritten:

$$(\min_i \boldsymbol{w}^T \boldsymbol{s}^{[i]})^2 \geq \boldsymbol{w}^T \boldsymbol{F} \boldsymbol{w} \quad \forall \boldsymbol{w} \in \mathbb{R}^n, |\boldsymbol{w}| = 1, \quad \text{equality only for } \boldsymbol{w} \propto \boldsymbol{e}_k \tag{5.29}$$

Further, notice that we can swap the minima in the inequality for a maxima. We can do this because if the inequality is satisfied for a particular value of $\boldsymbol{w}$ then the following inequality is satisfied for $-\boldsymbol{w}$:

$$(\max_i \boldsymbol{w}^T \boldsymbol{s}^{[i]})^2 \geq \boldsymbol{w}^T \boldsymbol{F} \boldsymbol{w} \quad \forall \boldsymbol{w} \in \mathbb{R}^n, |\boldsymbol{w}| = 1, \quad \text{equality only for } \boldsymbol{w} \propto \boldsymbol{e}_k \tag{5.30}$$

Since eq. (5.29) is satisfied for all $\boldsymbol{w}$, and all $\boldsymbol{w}$ have their $-\boldsymbol{w}$ unit vector pairs, so is eq. (5.30). Similarly, if eq. (5.30) is satisfied for all unit vectors $\boldsymbol{w}$, eq. (5.29) is satisfied but using $-\boldsymbol{w}$. Hence the two are equivalent.

Sufficiency of Condition

First, let's show the convex hull enclosing E is a sufficient condition. Choose an arbitrary unit vector $\boldsymbol{w}$. Choose the point within E that maximises the dot product with $\boldsymbol{w}$:

$$\boldsymbol{y}_w = \arg\max_{\boldsymbol{y} \in E} \boldsymbol{w}^T \boldsymbol{y} \tag{5.31}$$

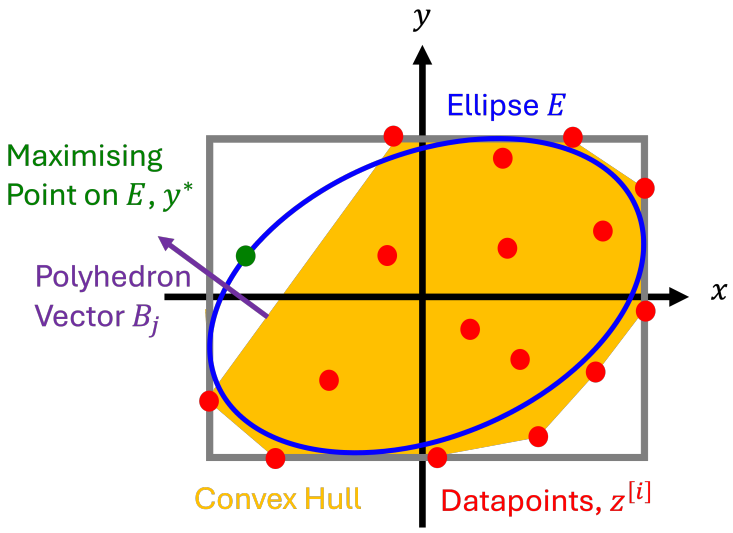


Figure 5.4: If the convex hull does not enclose the ellipse then you can always find a unit vector that breaks the inequality. In particular, this is true for the unit vector that defines the faces of the convex hull, $\boldsymbol{B}_j$.

We can find $\boldsymbol{y}_w$ by lagrange optimisation under the constraint that $\boldsymbol{y}^T\boldsymbol{F}^{-1}\boldsymbol{y} = 1$:

$$\boldsymbol{y}_w = \frac{\boldsymbol{F}\boldsymbol{w}}{\sqrt{\boldsymbol{w}^T\boldsymbol{F}\boldsymbol{w}}} \tag{5.32}$$

By the definition of E and its enclosure in the convex hull of the points we can write:

$$\boldsymbol{y}_w = \sum_i \lambda_i \boldsymbol{s}^{[i]} \quad \lambda_i > 0, \sum_i \lambda_i \leq 1 \tag{5.33}$$

Now:

$$\boldsymbol{w}^T\boldsymbol{y} = \sqrt{\boldsymbol{w}^T\boldsymbol{F}\boldsymbol{w}} = \sum_i \lambda_i \boldsymbol{w}^T\boldsymbol{s}^{[i]} \leq \max_i \boldsymbol{w}^T\boldsymbol{s}^{[i]} \sum_i \lambda_i = \max_i \boldsymbol{w}^T\boldsymbol{s}^{[i]} \tag{5.34}$$

Therefore, we get our our main result:

$$(\max_i \boldsymbol{w}^T\boldsymbol{s}^{[i]})^2 \geq \boldsymbol{w}^T\boldsymbol{F}\boldsymbol{w} \tag{5.35}$$

Further, since the only points at which E is allowed to touch the convex hull of the data are those at which the tangent to E is orthogonal to a unit vector, we can see that we can slightly increase the magnitude of $y = \boldsymbol{w}$ while staying within the convex hull. This turns the loose inequality $\geq$ into a strict one, $>$, as long as $\boldsymbol{w}$ is not a basis vector. If it is the equality is satisfied, since:

$$\boldsymbol{e}_k^T\boldsymbol{F}\boldsymbol{e}_k = (\min_i[s_k^{[i]}])^2 = (\min_i[\boldsymbol{e}_k^T\boldsymbol{s}^{[i]}])^2 \tag{5.36}$$

Necessity of Condition

Now we will show the necessity of this condition. We will do this by showing that if one of the points in E is not within the convex hull of the datapoints then we can find one of the inequalities in eq. (5.30) that is broken. .

First, get all the datapoints that for some unit vector $\boldsymbol{w}$ that maximise $\boldsymbol{w}^T\boldsymbol{s}^{[i]}$ over the dataset. Since for $\boldsymbol{w} \propto \boldsymbol{e}_k$ eq. (5.30) is an equality, we can just consider $\boldsymbol{w}$ that is not a basis direction. Construct the convex hull of these maximising points, the polyhedron P. You can equivalently characterise any polyhedra by a set of linear inequalities: $P = \{\boldsymbol{x} \in \mathbb{R} : \boldsymbol{B}\boldsymbol{x} < \boldsymbol{b}\}$. Denote with $\boldsymbol{B}_j$ the jth row of $\boldsymbol{B}$. We can always choose to write the inequalities in such a way that the rows of $\boldsymbol{B}$ are unit length, by appropriately rescaling each b_j. Assume we have done this.

Assume the convex hull does not contain the entire ellipse. Therefore there is at least one point on the ellipse that breaks one of the inequalities defining the polyhedron. These points are charactersied by being both on the ellipse, $\boldsymbol{y}^T\boldsymbol{F}^{-1}\boldsymbol{y} = 1$, but at least one of the inequalities defining the polyhedron is broken: $\boldsymbol{B}_j^T\boldsymbol{y} > b_j$. Let's call the set of points on the ellipse for which this inequality is broken $\mathcal{B}_j = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{y}^T\boldsymbol{F}^{-1}\boldsymbol{y} = 1 \quad \text{and} \quad \boldsymbol{B}_j^T\boldsymbol{y} > b_j\}$.

Now, choose the element of $\mathcal{B}_j$ that maximises $\boldsymbol{B}_j^T\boldsymbol{y}$, call it $\boldsymbol{y}^*$. This will also be the element of E that maximises $\boldsymbol{B}_j^T\boldsymbol{y}$, which by the previous lagrange optimisation, eq. (5.32), has the following dot product: $\boldsymbol{B}_j^T\boldsymbol{y}^* = \sqrt{\boldsymbol{B}_j^T\boldsymbol{F}\boldsymbol{B}_j}$.

But now we derive a contradiction. This point is outside the polyhedron, $\boldsymbol{B}_j^T \boldsymbol{y} > b_j$, whereas all points inside the polyhedron, including all the datapoints, satisfy all inequalities, hence, $\boldsymbol{B}_j^T \boldsymbol{s}^{[i]} \leq b_j \forall \boldsymbol{s}^{[i]}$. But, $\boldsymbol{B}_j^T \boldsymbol{y}^* = \sqrt{\boldsymbol{B}_j^T \boldsymbol{F} \boldsymbol{B}_j} > b_j$, therefore:

$$\boldsymbol{B}_j^T \boldsymbol{s}^{[i]} \leq b_j < \sqrt{\boldsymbol{B}_j^T \boldsymbol{F} \boldsymbol{B}_j} \quad \forall \boldsymbol{s}^{[i]} \tag{5.37}$$

And hence we've found a unit vector $\boldsymbol{B}_i$ such that:

$$\max_i \boldsymbol{B}_j^T \boldsymbol{s}^{[i]} < \sqrt{\boldsymbol{B}_j^T \boldsymbol{F} \boldsymbol{B}_j} \quad \rightarrow \quad (\max_i \boldsymbol{B}_j^T \boldsymbol{s}^{[i]})^2 < \boldsymbol{B}_j^T \boldsymbol{F} \boldsymbol{B}_j \tag{5.38}$$

Hence one of the inequalities in *eq.* (5.30) is broken, showing the necessity.

5.A.3 Range-Independent Variables

We will now prove the corollary on range dependence, corollary 2.1, recapped here:

Corollary 4.1. *In the same setting as theorem 1 the optimal representation modularises if all sources are pairwise extreme-point independent, i.e. if*

$$\min_i \left[s_j^{[i]} \middle| s_{j'}^{[i]} \in \left\{ \max_{i'} s_{j'}^{[i']}, \min_{i'} s_{j'}^{[i']} \right\} \right] = \min_i s_j^{[i]} \tag{5.39}$$

for all $j, j' \in [d_s]^2$.

Proof. If the variables are extreme-point independent then we can take the min inside the sum in the definition of the mixed neuron bias:

$$z_n^{[i]} = \sum_{j=1}^{d_s} u_{nj} \bar{s}_j^{[i]} + \Delta_n = \sum_{j=1}^{d_s} u_{nj} \bar{s}_j^{[i]} - \min_i [\sum_{j=1}^{d_s} u_{nj} \bar{s}_j^{[i]}] = \sum_{j=1}^{d_s} \left(u_{nj} \bar{s}_j^{[i]} - \min_i [u_{nj} \bar{s}_j^{[i]}] \right) \tag{5.40}$$

Now we can compute the activity loss:

$$\langle (z_n^{[i]})^2 \rangle_i = \sum_{j=1}^{d_s} \langle \left(u_{nj} \bar{s}_j^{[i]} - \min_i [u_{nj} \bar{s}_j^{[i]}] \right)^2 \rangle_i + \sum_{j,j'=1, j \neq j}^{d_s} \langle \left(u_{nj} \bar{s}_j^{[i]} - \min_i [u_{nj} \bar{s}_j^{[i]}] \right) \left(u_{nj} \bar{s}_{j'}^{[i]} - \min_i [u_{nj} \bar{s}_{j'}^{[i]}] \right) \rangle_i \tag{5.41}$$

The cross terms at the end are strictly positive, and each of the squared terms on the left is the same or larger than its corresponding modular neuron, since either u_{nj} is positive and it is the same, or u_{nj} is negative and it is greater than or equal to the equivalent modular neuron. Therefore the mixed energy cost is larger than the modular energy cost for extreme-point independent variables, so the optimal representation should modularise. □

5.A.4 Extension to all Activity Norms

We now extend our theoretical results to other activity norms beyond L2. As shown in theorem theorem 3, the weight loss is minimised by orthogonalising the coding of different variables, so we can focus on the activity loss. We will show that, for all norms, modular solutions are optimal if the variables are range independent.

Consider a set of modular neurons:

$$\boldsymbol{z}_n^{[i]} = \begin{bmatrix} |u_{n1}|(\bar{s}_1^{[i]} - \min_j \bar{s}_1^{[j]}) \\ \vdots \\ |u_{nd_s}|(\bar{s}_{d_s}^{[i]} - \min_j \bar{s}_{d_s}^{[i]}) \end{bmatrix} \geq \mathbf{0} \tag{5.42}$$

And let's see if we could usefully mix these encodings together for any set of mixing coefficients $\boldsymbol{u}_n \in \mathbb{R}^{d_s}$:

$$z_n^{[i]} = \sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} + \Delta_n = \sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} - \min_i [\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]}] \geq 0 \tag{5.43}$$

If the variables are range-independent then we can move the min inside the sum:

$$\Delta_n = -\min_i [\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]}] = -\sum_{j=1}^{d_s} \min_i u_{nj}\bar{s}_j^{[i]} \tag{5.44}$$

Then the Lp norm is:

$$\langle |z_n^{[i]}|^p \rangle_i = \langle | \left(\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} + \Delta_n \right) |^p \rangle_i = \langle | \left(\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} - \min_i u_{nj}\bar{s}_j^{[i]} \right) |^p \rangle_i \tag{5.45}$$

Now using the convexity of the Lp norm:

$$\langle | \left(\sum_{j=1}^{d_s} u_{nj}\bar{s}_j^{[i]} - \min_i u_{nj}\bar{s}_j^{[i]} \right) |^p \rangle_i \geq \sum_{j=1}^{d_s} \langle |u_{nj}\bar{s}_j^{[i]} - \min_i u_{nj}\bar{s}_j^{[i]}|^p \rangle_i \tag{5.46}$$

with equality only if p is equal to 1, or if at most one of the entries of $\boldsymbol{u}$ are 0.

Finally, since we orient our variables such that $|\min_i \bar{s}_j^{[i]}| < \max_i \bar{s}^{[i]}$, without loss of generality,:

$$\langle |u_{nj}\bar{s}_j^{[i]} - \min_i u_{nj}\bar{s}_j^{[i]}|^p \rangle_i \geq \langle |u_{nj}|^p \langle |\bar{s}_j^{[i]} - \min_i \bar{s}_j^{[i]}|^p \rangle_i \tag{5.47}$$

So we get our final result that the mixed activity loss is greater than or equal to the modular activity loss if the variables are range independent:

$$\langle |z_n^{[i]}|^p \rangle_i \geq \sum_{j=1}^{d_s} |u_{nj}|^p \langle |\bar{s}_j^{[i]} - \min_i \bar{s}_j^{[i]}|^p \rangle_i = \langle |\boldsymbol{z}_n^{[i]}|^p \rangle_i \tag{5.48}$$

So, as long as $p > 1$, the activity loss is lowered by modularising the representation of range-independent variables. Further, since the weight loss is also minimised, the optimal solution will be modular.

Two caveats: first, in the particular case of the L1 norm all that we can guarantee is that the activity loss of a modular solution is equal to that of a mixed solution. This means that, even in the case of range-independent variables, there might be orthogonal but mixed solutions that are equally good as the modular solution. For $p > 1$ this caveat is not needed.

Second, this is a set of sufficient conditions, range-independent variables are optimally encoded modularly. There will be other nearly range-independent variables that are also optimally encoded modularly, just as we found for the L2 case.

5.A.5 Extension to Multidimensional Variables

Many variables in the real world, such as angles on a circle, are not 1-dimensional. Ideally our theory would tell us when such multi-dimensional sources were modularised. Unfortunately, precise necessary and sufficient conditions are not an easy step from our current theory because our proof strategies make extensive use of the optimal modular encoding. For one-dimensional sources this is very easy to find, simply align each source correctly in a neuron, and make the representation positive. Finding similar optimal modular solutions for multidimensional sources is much harder, potentially as hard as simply solving the optimisation problem.

However, we can make progress on a simpler problem: showing that range-independence is a sufficient condition for a set of multi-dimensional sources to modularise. Consider two multidimensional sources $\boldsymbol{x}^{[i]} \in \mathbb{R}^{d_x}$ and $\boldsymbol{y}^{[i]} \in \mathbb{R}^{d_y}$, and an encoding in which they are mixed in single neurons:

$$z_n^{[i]} = \sum_{j=1}^{d_x} u_{nj} x_j^{[i]} + \sum_{j=1}^{d_y} v_{nj} y_j^{[i]} + \Delta_n \tag{5.49}$$

Whatever choice of coefficient vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$, or activity norm (as long as $p > 1$), we can consider breaking this mixed coding into two modular neurons. Define new variables $a^{[i]} = \sum_{j=1}^{d_x} u_{nj} x_j^{[i]}$ and $b^{[i]} = \sum_{j=1}^{d_y} v_{nj} y_j^{[i]}$. If the two multidimensional sources are range-independent, then so too are a and b. Define the modular representation:

$$\boldsymbol{z}_n^{[i]} = \begin{bmatrix} a^{[i]} - \min_j a^{[j]} \\ b^{[i]} - \min_j b^{[j]} \end{bmatrix} \tag{5.50}$$

All of our previous arguments apply to argue that this modular encoding will be preferred if the multidimensional variables are range-independent.

We just have to show that the weight loss is also reduced by modularising multidimensional sources, a small extension of theorem 3, which we shall now show. Write a modular code as:

$$\boldsymbol{z}^{[i]} = \boldsymbol{U}\boldsymbol{x}^{[i]} - \min_j[\boldsymbol{U}\boldsymbol{x}^{[j]}] + \boldsymbol{V}\boldsymbol{y}^{[i]} - \min_j[\boldsymbol{V}\boldsymbol{y}^{[j]}] \tag{5.51}$$

With the encoding of $\boldsymbol{x}$ and $\boldsymbol{y}$ spanning two different spaces: $\boldsymbol{U}^T\boldsymbol{V} = \boldsymbol{0}$.

Then define the minimal L2 norm readout weight matrix using the pseudoinverse:

$$\boldsymbol{W}_{\text{out}} = \begin{bmatrix} \boldsymbol{U}^\dagger \\ \boldsymbol{V}^\dagger \end{bmatrix} \tag{5.52}$$

Such that $\boldsymbol{W}_{\text{out}}\boldsymbol{z}^{[i]} = \begin{bmatrix} \boldsymbol{x}^{[i]} \\ \boldsymbol{y}^{[i]} \end{bmatrix}$ up to some constant offset that can be removed by the bias term in the readout. Now consider mixing the representation of the two multidimensional sources using an orthogonal matrix that preserves the encoding size of each variable, $\boldsymbol{O}$:

$$\boldsymbol{z}^{[i]} = \boldsymbol{U}\boldsymbol{x}^{[i]} + \boldsymbol{O}\boldsymbol{V}\boldsymbol{y}^{[i]} \qquad \boldsymbol{W}_{\text{out}} = \boldsymbol{W}_{\text{out}} = \begin{bmatrix} \boldsymbol{U}^\dagger + \boldsymbol{A} \\ \boldsymbol{V}^\dagger\boldsymbol{O}^T + \boldsymbol{B} \end{bmatrix} \tag{5.53}$$

Then in the subspace spanned by the columns of $\boldsymbol{U}$ $\boldsymbol{W}_{\text{out}}$ must be the same as $\boldsymbol{U}^\dagger$. However, in order to offset the newly aligned matrix $\boldsymbol{O}\tilde{\boldsymbol{V}}$ there must be an additional component orthogonal, $\boldsymbol{A}$:

$$\begin{aligned} \boldsymbol{W}_{out}\boldsymbol{U} = \boldsymbol{U}^\dagger\boldsymbol{U} + \boldsymbol{A}\boldsymbol{U} = \mathbb{I} \qquad \boldsymbol{A}\boldsymbol{U} = \boldsymbol{0} \\ \boldsymbol{W}_{out}\boldsymbol{O}\boldsymbol{V} = \boldsymbol{U}^\dagger\boldsymbol{O}\boldsymbol{V} + \boldsymbol{A}\boldsymbol{O}\boldsymbol{V} = \boldsymbol{0} \qquad \boldsymbol{A}\boldsymbol{O}\boldsymbol{V} = -\boldsymbol{U}^\dagger\boldsymbol{O}\boldsymbol{V} \end{aligned} \tag{5.54}$$

So $\boldsymbol{A}$ and $\boldsymbol{U}^\dagger$ live in orthogonal subspaces, and $\boldsymbol{A} \neq 0$. The same applies to $\boldsymbol{V}$ and $\boldsymbol{B}$, therefore:

$$||\boldsymbol{W}_{out}||_F^2 = ||\boldsymbol{U}^\dagger||_F^2 + ||\boldsymbol{V}^\dagger||_F^2 + ||\boldsymbol{A}||_F^2 + ||\boldsymbol{B}||_F^2 \geq ||\boldsymbol{U}^\dagger||_F^2 + ||\boldsymbol{V}^\dagger||_F^2 \tag{5.55}$$

And hence orthogonalising the subspaces reduces the loss in this case as well.

Combining these two, the optimal representation of range-independent multi-dimensional variables for activity losses using an Lp norm for $p > 1$ is modular.

Chapter 6

A Recurrent Modularisation Theory: How Grid Cells got their Modules

Thus far our work on modularisation has fallen into the same trap as the Efficient Coding Hypothesis: we have focused on passive feedforward representations, rather than the implementations of interesting ongoing computations. In this section we make a first step towards extending these kinds of analyses to recurrent settings, allowing us to make statements about how they should decompose their functions. We show that similar ideas hold: range independent variables are encoded in modular recurrent blocks and we use this to understand the modularisation of grid cells, as observed in part I. Further, we show that these ideas empirically generalise to nonlinear settings. We are excited by the prospect that an elaborated form of this theory might tell us how an RNN chooses to decompose its computations.

6.1 Modularisation in Biologically Inspired Recurrent Networks

Compared to feedforward networks, recurrent neural networks (RNNs) are often a much more natural setting for neuroscience. Excitingly, the core ideas of our analysis for linear autoencoders also apply to recurrent dynamical formulations, and similarly extend to experiments with nonlinear networks.

6.1.1 Linear RNNs

Linear sinusoidal regression. Linear dynamical systems can only autonomously implement exponentially growing, decaying, or stable sinusoidal functions. We therefore study linear RNNs with biological constraints trained to model stable sinusoidal signals at certain frequencies:

$$\boldsymbol{z}(t+\delta t) = \boldsymbol{W}_{\text{rec}}\boldsymbol{z}(t) + \boldsymbol{b}_{\text{rec}}, \quad \boldsymbol{W}_{\text{out}}\boldsymbol{z}(t) = \begin{bmatrix}\cos(\omega_1 t + \phi_1)\\ \cos(\omega_2 t + \phi_2)\end{bmatrix}, \quad \boldsymbol{z}(t) \geq \boldsymbol{0}. \tag{6.1}$$

We study the optimal nonnegative, efficient, recurrent representations, $\boldsymbol{z}_t$, and show that whether the representations of the two frequencies within $\boldsymbol{z}_t$ modularises depends on their ratio. We prove and verify empirically that if one frequency is an integer multiple of the other the encodings mix, whereas if their ratio is irrational they should modularise. Further, we show empirically, and prove in most settings, that rational non-harmonic ratios should modularise (section 6.A). The intuition for this result is much the same as the linear autoencoding setting: the natural notions of sources are the signals $(\cos(\omega_1 t), \sin(\omega_1 t), \cos(\omega_2 t), \sin(\omega_2 t))$. Using these sources, we must simply ask: does their support allow for a reduction in activity energy via mixing? In fig. 6.1a, we visualise the source support for the three prototypical relationships between ω_1 and ω_2: irrational, rational (but not harmonic), and harmonic. Results of neural network verifications are in fig. 6.1b (details section 6.B). In the irrational case, the source support is essentially rectangular, so the model modularises. In the harmonic case, large chunks of various corners are missing from the source support, so the model mixes. In the rational case, even though the source support is quite sparse, the corners are sufficiently present such that modularising is still optimal.

Modularisation of grid cells. We now show that these spectral ideas can explain modules of grid cells. Grid cells are neurons in the mammalian entorhinal cortex that fire in a hexagonal lattice of positions (Hafting et al., 2005). They come in groups, called modules; grid cells within the same module have receptive fields (firing patterns) that are translated copies of the same lattice, and different modules are defined by their different lattice (H. Stensola et al., 2012). Current theories like the one we developed in the previous chapter suggest that the grid cell system can be modelled as a RNN with activations built from linear combinations

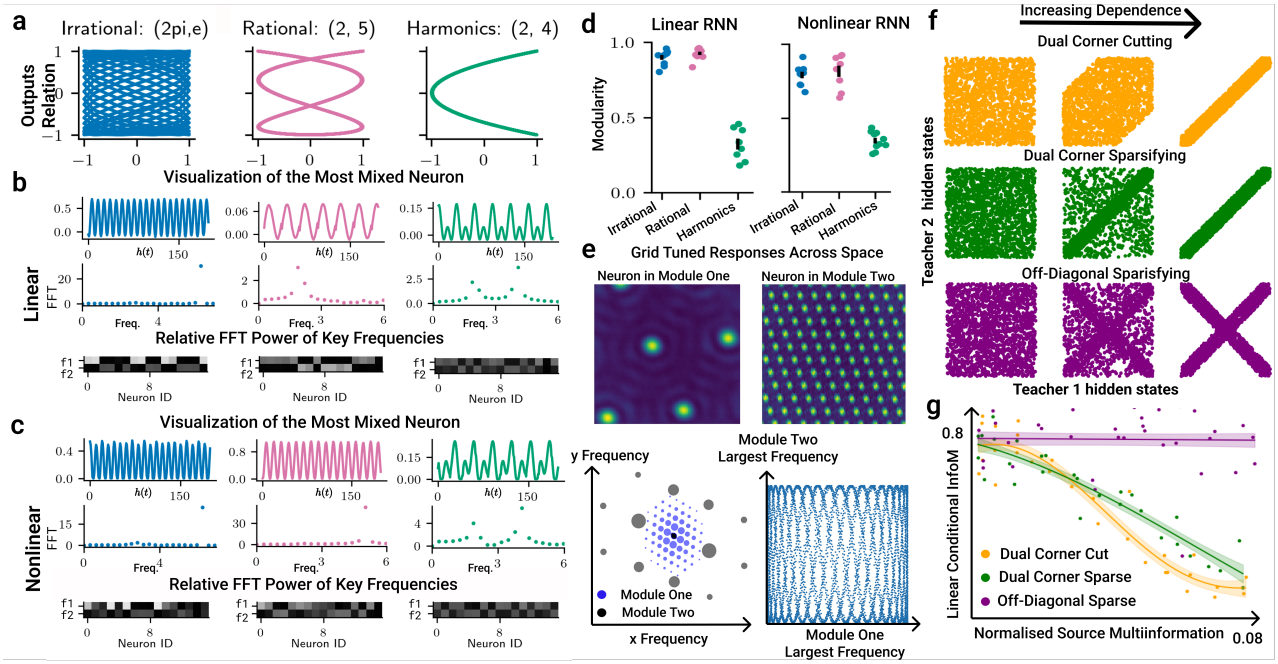


Figure 6.1: a) Source support visualisations: $\cos(\omega_1 t)$ vs. $\cos(\omega_2 t)$. b) The activity of the most mixed neuron in an optimal solution, the same neuron's Fourier spectrum, and the population's Fourier spectrum in the two task frequencies show clear modularisation in the irrational and rational cases, and clear mixing the in the harmonic case. c) Same as (b), but with *nonlinear* RNNs. d) Modularity of RNNs trained on 10 frequency pairs. e) Top: Two optimal grid cells from part I in different modules. Bottom: The modularisation of these two lattices can be understood from the range-independence of their constituent frequencies, as shown in the joint distribution of the most significant frequency in each module. f) Plot of joint distribution of hidden state activity for two teacher RNNs neurons as we increase the dependence in three different ways. g) Trends of modularity scores of student RNN in three different cases qualitatively agree with our theory.

of frequencies part I, just like the linear RNNs considered here. Importantly, these grid cell theories use the same biological constraints as in this work and show that modules form because the optimal code contains non-harmonically related frequencies that are encoded in different neurons (fig. 6.1e top). This modularisation can now be theoretically justified in our framework, since non-harmonic frequencies are range-independent and so should be modularised (fig. 6.1e bottom).

6.1.2 Nonlinear RNNs

Mixed sinusoidal regression for nonlinear RNNs. To test how our ideas generalise beyond linear networks, we train nonlinear ReLU RNNs with biologically inspired constraints to perform a frequency mixing task (details section 6.B). We provide a pulse input $P_\omega(t) = \mathbb{I}\left[\mathrm{mod}_\omega(t) = 0\right]$ at two frequencies, and the network has to output the resulting "beats" and "carrier" signals:

$$\boldsymbol{z}(t+\Delta t) = \mathrm{ReLU}\left(\boldsymbol{W}_{\mathrm{rec}}\boldsymbol{z}(t) + \boldsymbol{W}_{\mathrm{in}}\begin{bmatrix}P_{\omega_1}(t)\\ P_{\omega_2}(t)\end{bmatrix} + \boldsymbol{b}_{\mathrm{rec}}\right), \quad \begin{bmatrix}\cos([\omega_1-\omega_2]t)\\ \cos([\omega_1+\omega_2]t)\end{bmatrix} = \boldsymbol{W}_{\mathrm{out}}\boldsymbol{z}_t + \boldsymbol{b}_{\mathrm{out}}. \tag{6.2}$$

Results are in fig. 6.1c. Identical range-dependence properties but applied to the frequencies $\omega_1 - \omega_2$ and $\omega_1 + \omega_2$ determine whether or not the network modularises: irrational, range-independent frequencies modularise; harmonics, with their large missing corners, mix; and other rationally related frequencies are range-dependent but no sufficient corner is missing, so they modularise.

Modularisation in nonlinear teacher-student distillation. To test our predictions of when RNNs modularise, but in settings more realistic than pure frequencies, we generate training data trajectories from randomly initialised teacher RNNs with tanh activation function, and then train student RNNs (with a ReLU activation function) on these trajectories. The student's representation is constrained to be nonnegative (via its ReLU) and has its activity and weights regularised (see section 6.B.2 for details). Using carefully chosen inputs at each timestep, we are able to precisely control the teacher RNN hidden state activity (i.e., the source distribution). This allows us to change correlations/statistical independence of the hidden states, while either maintaining or breaking range independence. We consider three settings (fig. 6.1f). First, when statistical and range independence are progressively broken (in orange). Second, where statistical, and to a lesser extent range, independence gets progressively broken (in green). Third, where only statistical independence gets progressively

broken (in purple). We observe that, in line with our theory, preserving the corner points preserves modularity (fig. 6.1g purple), while removing them breaks it (orange). Further, spasifying the corners breaks modularity, but more slowly than removing corners (green).

6.1.3 Conclusion

As such, it seems like the key ideas of our analysis generalise to recurrent networks, including in limited nonlinear settings. This helps us to explain modularity patterns in grid cells, and will be useful as we turn to prefrontal recordings in the next chapter part III.

Technical Appendicies: Recurrent Modularisation

6.A Mathematical Analysis for Positive Linear Recurrent Networks

Here we mathematically study networks implementing simple frequency outputs. Linear dynamical systems can only produce mixtures of decaying or growing sinusoids, so we therefore ask the RNN to produce different frequency outputs via an affine readout:

$$\boldsymbol{W}_{\text{out}}\boldsymbol{g}(t) + \boldsymbol{b}_{\text{out}} = \begin{bmatrix}\cos(\omega_1 t)\\ \cos(\omega_2 t)\end{bmatrix} \tag{6.3}$$

Then we'll assume the internal dynamical structure is a standard linear RNN:

$$\boldsymbol{W}_{\text{rec}}\boldsymbol{g}(t) + \boldsymbol{b} = \boldsymbol{g}(t + \Delta t) \tag{6.4}$$

We will study this two frequency setting and ask when the representation learns to modularise the two frequencies. These results could be generalised to multiple variables as in section 5.A.5. Again, our representation must be non-negative, $\boldsymbol{g}(t) \geq \boldsymbol{0}$, and we minimise the following energy loss:

$$\mathcal{L} = \langle ||\boldsymbol{g}(t)||^2 \rangle + \lambda_W ||\boldsymbol{W}_{\text{rec}}||_F^2 + \lambda_R ||\boldsymbol{W}_{\text{out}}||_F^2 \tag{6.5}$$

We will show that if the RNN solves the task for infinite time, the optimal representation contains activity cycling at only the two frequencies, ω_1 and ω_2. Further we'll show that these two parts should modularise if the two frequencies are irrational ratios of one another, and should mix if one frequency is an integer multiple of the other, i.e. the frequencies are harmonics. Finally, we conjecture, show empircally, and present proof in most cases, that non-harmonic rational frequency ratios should modularise. If the task runs for finite time then a smudge factor would have to be introduced: the frequencies could be usefully mixed at near integer multiples, and how close you have to be to mix is governed by the length of the task.

To show this we will first study the weight losses, and show that they are minimised if each frequency is modularised from the others. We will then study representations containing only two frequencies, and show it is the frequency ratio that governs whether they can be usefully mixed. Finally, we will combine these analyses; as in section 5.A, when all losses agree on their minima, the representation modularises, else it mixes.

6.A.1 Structure of Representation - Act I

Due to the autonomous linear system architecture, the representation can only contain a linear mixture of growing, decaying, or stable frequencies, and a constant. The growing and decaying modes cannot help you solve the task (since if they do at one point in time, they later won't), and will either cost infinite energy (the growing modes) or will cost some energy without helping to reduce the bias (since, again, the bias has to be set large enough such that after the decaying mode has disappeared the representation is still positive). Hence,

$$\boldsymbol{g}(t) = \sum_{i=1}^{F} \boldsymbol{a}_i \cos(\omega_i t) + \boldsymbol{b}_i \sin(\omega_i t) + \boldsymbol{b}_0 \tag{6.6}$$

Where F is smaller than half the number of neurons to ensure the autonomous linear system can propagate each frequency (each frequency 'uses' 2 dimensions, as will be obvious from section 6.A.3.

6.A.2 Readout Loss

The readout loss is relatively easy, again create some capitalised pseudoinverse vectors $\{\boldsymbol{A}_i, \boldsymbol{B}_i\}_{i=1}^F$ defined by being the min-norm vectors with the property that $\boldsymbol{A}_i^T \boldsymbol{a}_j = \delta_{ij}$ and $\boldsymbol{A}_i^T \boldsymbol{b}_j = 0$, and the same for $\boldsymbol{B}_i$. Then the min-norm readout matrix is:

$$\boldsymbol{W}_{\text{out}} = \begin{bmatrix}\boldsymbol{A}_1^T\\ \boldsymbol{A}_2^T\end{bmatrix} \tag{6.7}$$

And the readout loss is:

$$|\boldsymbol{W}_{\text{out}}|_F^2 = \text{Tr}[\boldsymbol{W}_{\text{out}}\boldsymbol{W}_{\text{out}}^T] = |\boldsymbol{A}_1|^2 + |\boldsymbol{A}_2|^2 \tag{6.8}$$

Each vector has the following form, $\boldsymbol{A}_i = \frac{1}{\boldsymbol{a}_i^2}\boldsymbol{a}_i + \boldsymbol{a}_{i,\perp}$, where $\boldsymbol{a}_{i,\perp}$ is orthogonal to $\boldsymbol{a}_i$ and is included to ensure the correct orthogonality properties hold. So, for a fixed encoding size, the readout loss is minimised if $\boldsymbol{a}_{i,\perp} = 0$.

This occurs when the encodings are orthogonal (i.e. $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are orthogonal from one another, all other $\boldsymbol{a}_i$ vectors, and from each of the $\boldsymbol{b}_i$ vectors). This happens if the two frequencies are modularised from one another, and additionally the sine and cosine vectors for each frequency are orthogonal. I.e. a modularised solution with this property has the minimal readout weight loss for a given encoding size.

6.A.3 Recurrent Loss without Bias

First we consider the slightly easier case where there is no bias in the recurrent dynamics:

$$\boldsymbol{W}\boldsymbol{g}(t) = \boldsymbol{g}(t+1) \tag{6.9}$$

Then we will write down a convenient decomposition of the min-norm $\boldsymbol{W}$. Call the matrix of stacked coefficient vectors, $\boldsymbol{X}$:

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{b}_1 & \dots & \boldsymbol{a}_F & \boldsymbol{b}_F & \boldsymbol{b}_0\end{bmatrix} \tag{6.10}$$

Similarly, call the matrix of stacked normalised psuedo-inverse vectors $\boldsymbol{X}^\dagger$:

$$\boldsymbol{X}^\dagger = \begin{bmatrix}\boldsymbol{A}_1 & \boldsymbol{B}_1 & \dots & \boldsymbol{A}_F & \boldsymbol{B}_F & \boldsymbol{B}_0\end{bmatrix} \tag{6.11}$$

And finally create an ideal rotation matrix:

$$\begin{aligned}\boldsymbol{R} &= \begin{bmatrix} \cos(\omega_1\Delta t) & -\sin(\omega_1\Delta t) & \dots & 0 & 0 & 0 \\ \sin(\omega_1\Delta t) & \cos(\omega_1\Delta t) & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \cos(\omega_F\Delta t) & -\sin(\omega_F\Delta t) & 0 \\ 0 & 0 & \dots & \sin(\omega_F\Delta t) & \cos(\omega_F\Delta t) & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{R}_1 & 0 & \dots & 0 & 0 \\ 0 & \boldsymbol{R}_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \boldsymbol{R}_F & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}\end{aligned} \tag{6.12}$$

Where each $\boldsymbol{R}_i$ is a 2×2 rotation matrix at frequency ω_i. Now:

$$\boldsymbol{W} = \boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^{\dagger,T} \tag{6.13}$$

We can then calculate the recurrent weight loss:

$$|\boldsymbol{W}|_F^2 = \text{Tr}[\boldsymbol{W}\boldsymbol{W}^T] = \text{Tr}[\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^{\dagger,T}\boldsymbol{X}^\dagger\boldsymbol{R}^T] \tag{6.14}$$

$\boldsymbol{X}^T\boldsymbol{X}$ is a symmetric $2F+1 \times 2F+1$ positive-definite matrix, and its inverse is $\boldsymbol{X}^{\dagger,T}\boldsymbol{X}^\dagger$, another symmetric positive-definite matrix. To see that these matrices are inverses of one another perform the singular value decomposition, $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$, and $\boldsymbol{X}^\dagger = \boldsymbol{U}\Sigma^{-1}\boldsymbol{V}^T$. Then $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\Sigma^2\boldsymbol{V}^T$ and $\boldsymbol{X}^{\dagger,T}\boldsymbol{X}^\dagger = \boldsymbol{V}\Sigma^{-2}\boldsymbol{V}^T$, which are clearly inverses of one another.

Introducing a new variable, $\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}$:

$$|\boldsymbol{W}|_F^2 = \text{Tr}[\boldsymbol{Y}\boldsymbol{R}\boldsymbol{Y}^{-1}\boldsymbol{R}^T] \tag{6.15}$$

We then use the following trace inequality, from Ruhe (Ruhe, 1970). For two positive semi-definite symmetric matrices, $\boldsymbol{E}$ and $\boldsymbol{F}$, with ordered eigenvalues, $e_1 \geq \ldots \geq e_n \geq 0$ and $f_1 \geq \ldots \geq f_n \geq 0$

$$\text{Tr}[\boldsymbol{E}\boldsymbol{F}] \geq \sum_{i=1}^{n} e_i f_{n-i+1} \tag{6.16}$$

Now, since $\boldsymbol{R}\boldsymbol{Y}^{-1}\boldsymbol{R}^T$ and $\boldsymbol{Y}^{-1}$ are similar matrices, they have the same eigenvalues, and $\boldsymbol{Y}^{-1}$ is the inverse of $\boldsymbol{Y}$ so its eigenvalues are the inverse of those of $\boldsymbol{Y}$. Therefore:

$$|\boldsymbol{W}|_F^2 = \text{Tr}[\boldsymbol{Y}\boldsymbol{R}\boldsymbol{Y}^{-1}\boldsymbol{R}^T] \geq \sum_{i=1}^{2F+1} \frac{\lambda_i}{\lambda_i} = 2F+1 \tag{6.17}$$

Then we can show that this lower bound on the weight loss is achieved when the coefficient vectors are orthogonal, hence making the modular solution optimal. If all the coefficient vectors are orthogonal then $\boldsymbol{Y}$ and $\boldsymbol{Y}^{-1}$ are diagonal, so they commute with any matrix, and:

$$|\boldsymbol{W}|_F^2 = \text{Tr}[\boldsymbol{Y}\boldsymbol{R}\boldsymbol{Y}^{-1}\boldsymbol{R}^T] = \text{Tr}[\boldsymbol{Y}\boldsymbol{Y}^{-1}\boldsymbol{R}^T\boldsymbol{R}] = \text{Tr}[\mathbb{I}_{2F+1}] = 2F+1 \tag{6.18}$$

6.A.4 Recurrent Loss with Bias

Now we return to the case of interest:

$$\boldsymbol{W}\boldsymbol{g}(t) + \boldsymbol{b} = \boldsymbol{g}(t+1) \tag{6.19}$$

Our energy loss, equation 6.5, penalises the size of the weight matrix $|\boldsymbol{W}|_F^2$ and not the bias. Therefore, if we can make $\boldsymbol{W}$ smaller by assigning some of its job to $\boldsymbol{b}$ then we should. We can do this by setting $\boldsymbol{b}_0 = \boldsymbol{b}$ (recall the definition of $\boldsymbol{b}_0$ from equation 6.6) and constructing the following, smaller, min-norm weight matrix:

$$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{b}_1 & \dots & \boldsymbol{a}_F & \boldsymbol{b}_F\end{bmatrix} \begin{bmatrix}\boldsymbol{R}_1 & 0 & \dots & 0 \\ 0 & \boldsymbol{R}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{R}_F\end{bmatrix} \begin{bmatrix}\boldsymbol{A}_1^T \\ \boldsymbol{B}_1^T \\ \vdots \\ \boldsymbol{A}_F^T \\ \boldsymbol{B}_F^T\end{bmatrix} = \hat{\boldsymbol{X}}\hat{\boldsymbol{R}}\hat{\boldsymbol{X}}^\dagger \tag{6.20}$$

Using the definitions in the previous section. This slightly complicates our previous analysis because now $\hat{\boldsymbol{X}}^{\dagger,T}\hat{\boldsymbol{X}} = \hat{Y}^\dagger$ is not the inverse of $\hat{\boldsymbol{X}}^T\hat{\boldsymbol{X}} = \hat{\boldsymbol{Y}}$. If $\boldsymbol{b}$ is orthogonal to all the vectors $\{\boldsymbol{a}_i, \boldsymbol{b}_i\}_{i=1}^F$, then it is the inverse, and the previous proof that modularity is an optima goes through.

Fortunately, it is easy to generalise to this setting. $\boldsymbol{X}^\dagger$ is not quite the pseudoinverse of $\boldsymbol{X}^\dagger$ because its vectors have to additionally be orthogonal to $\boldsymbol{b}$. This means we can break down each of the vectors into two components, for example:

$$\boldsymbol{A}_1 = \hat{\boldsymbol{A}}_1 + \hat{\boldsymbol{A}_{1,\perp}} \tag{6.21}$$

The first of these vectors is the transpose of the equivalent row of the pseudoinverse of $\hat{\boldsymbol{X}}$, it lives in the span of the vectors $\{\boldsymbol{a}_i, \boldsymbol{b}_i\}_{i=1}^F$. The second component is orthogonal to this span and ensures that $\boldsymbol{A}_1^T\boldsymbol{b} = 0$. Hence, the previous claim. If $\boldsymbol{b}$ is orthogonal to the span of $\{\boldsymbol{a}_i, \boldsymbol{b}_i\}_{i=1}^F$, this is the standard pseudoinverse and the previous result goes through.

We can express the entire $\hat{\boldsymbol{X}}^\dagger$ matrix in these two components:

$$\hat{\boldsymbol{X}}^\dagger = \hat{\boldsymbol{X}}_0^\dagger + \hat{\boldsymbol{X}}_\perp^\dagger \tag{6.22}$$

Then:

$$\hat{\boldsymbol{Y}}^\dagger = (\hat{\boldsymbol{X}}_0^\dagger + \hat{\boldsymbol{X}}_\perp^\dagger)^T(\hat{\boldsymbol{X}}_0^\dagger + \hat{\boldsymbol{X}}_\perp^\dagger) = \hat{\boldsymbol{X}}_0^{\dagger,T}\hat{\boldsymbol{X}}_0^\dagger + \hat{\boldsymbol{X}}_\perp^{\dagger,T}\hat{\boldsymbol{X}}_\perp^\dagger = \hat{\boldsymbol{Y}}^{-1} + \hat{\boldsymbol{Y}}_\perp^\dagger \tag{6.23}$$

Both of these new matrices are positive semi-definite matrices, since they are formed by taking the dot product of a set of vectors. Then:

$$|\boldsymbol{W}|_F^2 = \mathrm{Tr}[\hat{\boldsymbol{Y}}\hat{\boldsymbol{R}}\hat{\boldsymbol{Y}}^{-1}\hat{\boldsymbol{R}}^T] + [\hat{\boldsymbol{Y}}\hat{\boldsymbol{R}}\hat{\boldsymbol{Y}}_\perp^\dagger\hat{\boldsymbol{R}}^T] \tag{6.24}$$

Now, the first term is greater than or equal than $2F$, as in the previous setting. And since $\hat{\boldsymbol{Y}}$ and $\hat{\boldsymbol{Y}}_\perp^\dagger$ are positive semi-definite, and so therefore is $\hat{\boldsymbol{R}}\hat{\boldsymbol{Y}}_\perp^\dagger\hat{\boldsymbol{R}}^T$, this second term is greater than or equal to 0. Hence, $|\boldsymbol{W}|_F^2 \geq 2F$, and orthogonal encodings achieve this bound, therefore it is an optimal solution according to the weight loss.

6.A.5 Mixing Two Frequencies

Now, as a simplified version of the full activity loss, we ask when should two frequencies mix according to the activity loss. Therefore, consider a mixed frequency neuron:

$$g_i(t) = (\boldsymbol{a}_1)_i \cos(\omega_1 t) + (\boldsymbol{b}_1)_i \sin(\omega_1 t) + (\boldsymbol{a}_2)_i \cos(\omega_2 t) + (\boldsymbol{b}_2)_i \sin(\omega_2 t) + b_n \tag{6.25}$$

By rescaling and shifting time, we can rewrite all such mixed neurons in a simpler form:

$$g_i(t) = \alpha_i \cos(t) + \beta_i \cos(\omega t + \phi) + b_i \quad \omega > 1 \tag{6.26}$$

Where ω is the ratio of the larger to the smaller frequencies. We will show that the activity of this mixed neuron is lower than its corresponding modular counterpart when ω is an integer, and higher if ω is irrational. Finally, we show empirically, and present partial proof, that the same is true of rational, non-integer ω (in the rational case $\omega = \frac{p}{q}$ for two integers p and q greater than 1 that don't share a common factor; we show that, except when either p or q is two, rational frequencies should modularise). Hence, we find the activity energy is minimised by modularising unless one of the frequencies is an integer multiple of the others. Further, since if the mixed neuron $g_i(t)$ is preferred over its modular counterpart, then so too are positively scaled versions, $\mu g_i(t)$, we will consider responses of the form:

$$g(t) = \cos(t) + \delta \cos(\omega t + \phi) + \Delta \quad \omega > 1 \tag{6.27}$$

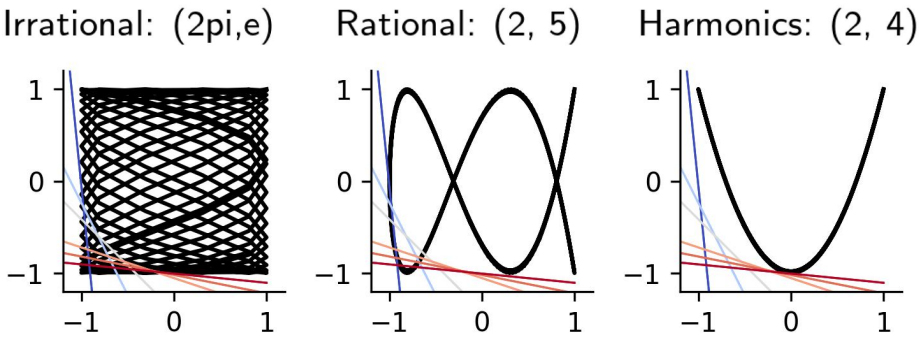


Figure 6.2: Schematics showing irrational and rational ratio case the data are range dependent beyond the modular-mixed boundary while in de-modularising harmonics case, the two periodic waves are range independent.

Irrational Frequencies Modularise If ω is irrational then, by Kronecker's theorem, you can find a value of t for which $\cos(t)$ and $\cos(\omega t + \phi)$ take any pair of values. This makes the two frequencies, among other things, extreme point independent, and therefore no mixing is better than modularising.

Even Integer Multiples Mix According to theorem 1, to modularise, for all δ and ϕ values:

$$\Delta > \sqrt{1+\delta^2} = 1 + \mathcal{O}(\delta^2) \tag{6.28}$$

We will therefore try to find a δ and ϕ which breaks this inequality. Consider $\delta < 0$ and very small and $\phi = 0$, then, writing $\omega = 2m$ for integer m:

$$\Delta = -\min_i[\cos(t) + \delta\cos(2mt)] \tag{6.29}$$

$\cos(t) - |\delta|\cos(2mt)$ takes its extreme values when t is an integer multiple of π, and the smallest it can be is $-1 + |\delta|$. Hence:

$$\Delta = 1 - \delta < 1 + \mathcal{O}(\delta^2) \tag{6.30}$$

This is smaller than the critical value, so the representation should mix, since at least one mixing inequality was broken.

Odd Integer Multiples Mix If ω is an odd integer then we can instead mix cosine with sine by choosing $\phi = \frac{\pi}{2}$:

$$g_i(t) = \cos(t) + \delta\sin(\omega t) + \Delta \tag{6.31}$$

Then the same argument goes through as above.

Other Rational Multiples Modularise Now consider $\omega = \frac{p}{q}$ for two integers p and $q \neq 1$. We will consider w.l.o.g. only the most simple fraction $\frac{p}{q}$, i.e. p and q share no factors, else simplify the fraction so they do. In this case, since neither is one, it is clear that either one or other of the integers must be at least 3. This will be useful later.

In this section we use the proof developed by maths stack exchange user Alex Ravsky (Ravsky, 2025). Consider the function:

$$f(t) = \cos(t) + \delta\cos(\omega t + \phi) \qquad \Delta = -\min_t[f(t)] \tag{6.32}$$

Consider $t = (2k+1)\pi$ for integers k, making $\cos(t) = -1$. Our expression is then:

$$f(t) = -1 + \delta\cos(2\pi\omega k + \psi) \tag{6.33}$$

where $\psi = \omega\pi + \phi$. We know that, for any particular timepoint, t^*, the minimum is smaller than the function: $-\min_t[f(t)] \geq -f(t*)$. Therefore we will search amongst the set of k values for a timepoint such that $-f(t*) \geq \sqrt{1+\delta^2}$. If we can find one then we know the minima will be small enough, $\Delta \geq \sqrt{1+\delta^2}$, and the frequencies will modularise.

Now, since p and q don't share any factors, as k takes different integer values, the quantity $2\pi\omega k$ modulo 2π will reach all the values in the set $\{0, \frac{2\pi}{q}, \frac{4\pi}{q}, ..., \frac{2\pi(q-1)}{q}\}$. This means that inside any fixed interval of length $\frac{2\pi}{q}$ there will be a number equal to $2\pi\omega k + \psi - 2l\pi$ for some choice of integers k and l.

In particular, when $\delta > 0$, let's choose the interval $I = [\pi + \frac{-\pi}{q}, \pi + \frac{\pi}{q}]$. Since we know that, for some choice of k and l, we can put t^* within this interval, we can upper-bound the function at this point with the evaluation of the cosine at its maximal value within the interval I:

$$f(t^*) = -1 + \delta\cos(2\pi\omega k + \psi - 2l\pi) \leq -1 + \max_{t' \in I} \delta\cos(t') = -1 + \delta\cos(\pi + \frac{\pi}{q}) = -1 - \delta\cos(\frac{\pi}{q}) \tag{6.34}$$

If instead $\delta > 0$ consider the interval $I = [-\frac{\pi}{q}, \frac{\pi}{q}]$, and you can reach a similar conclusion:

$$f(t^*) \leq -1 - |\delta|\cos(\frac{\pi}{q}) \tag{6.35}$$

Therefore:

$$-\min_t[f(t)] \geq -f(t^*) \geq 1 + |\delta|\cos(\frac{\pi}{q}) \tag{6.36}$$

Assume $q \geq 3$ and $|\delta| < \frac{4}{3}$, then:

$$-\min_t[f(t)] \geq 1 + |\delta|\cos(\frac{\pi}{q}) \geq 1 + |\delta|\cos(\frac{\pi}{3}) = 1 + \frac{|\delta|}{2} \geq \sqrt{1+\delta^2} \tag{6.37}$$

Now, that covers a subset of the cases. To get another, switch the role of the two frequencies. Change variables to $\tau = \omega t + \phi$ and divide by $|\delta|$:

$$\frac{1}{|\delta|}f(\tau) = \frac{1}{|\delta|}\cos(\frac{1}{\omega}\tau - \frac{\phi}{\omega}) + \text{sign}(\delta)\cos(\tau) \tag{6.38}$$

If $\text{sign}(\delta) = -1$ change variables again to $\tau' = \pi - \tau$, so we can ignore it. This function can then be treated with the same logic as above but with the roles of ω and δ inverted. As such, under the conditions $p \geq 3$ and $|\delta| \geq \frac{4}{3}$, we can show the required argument:

$$-\min_\tau[\frac{1}{|\delta|}f(\tau)] \geq \sqrt{1 + \frac{1}{\delta^2}} \qquad -\min_\tau[f(\tau)] \geq \sqrt{1+\delta^2} \tag{6.39}$$

That covers another set of cases. If, however, either p or q is 2 then there are choices of δ for which neither argument works.

We show empirically in fig. 6.2 that for rational frequency pairs no ellipse bounds the support of the data, supporting our conjecture.

6.A.6 Structure of Representation - Act II

We saw that our representation takes the following form.

$$\boldsymbol{g}(t) = \sum_{i=1}^{F} \boldsymbol{a}_i \cos(\omega_i t) + \boldsymbol{b}_i \sin(\omega_i t) + \boldsymbol{b}_0 \tag{6.40}$$

Further, in our analysis of the weight losses, we saw that each additional frequency in the population increases the weight loss. To solve the task we only need two frequencies, so why would we have more than two? The only possibility is that by including additional frequencies we might save activity energy, at the cost of weight energy.

The labels function forces the representation to contain the two readout frequencies, ω_1 and ω_2. Let's think about the encoding of ω_1. We can make such a representation positive using a bias, but actually it i scheaper to use a small amount of a harmonic and do the rest of the job with the bias. Intuitively this is because the bias wastes firing energy pushing positive all firing rates, even if they were already very high. The second harmonic does a better job since it does not add as much bias at times when the harmonic is already positive.

By the activity analysis in the previous section the only frequencies that might be usefully included to the activity loss are those that are multiples of ω_1, since all others should be optimally modularised out, only contributing to the activity loss. Therefore, if we do have other frequencies, we know they will be harmonics of the two base frequencies. But that means our conclusions about when the whole population modularises are unchanged from when a pair of frequencies modularise: if two frequencies are irrationally related so are all their harmonics, and so the two sets should modularise. Conversely, if one frequency is a harmonic of the other, then all their harmonics are also harmonics of the base frequency, so they should mix. Finally, as always, the rational case is more complicated. Some of the harmonics of rationally related frequencies are harmonics of one another: we conjecture and see empirically that they never mix, but we cannot prove this.

6.B Details of Recurrent Numerical Experiments

6.B.1 Linear and Nonlinear Periodic Wave RNN Experiment Details

For the linear RNN setting we provided a periodic pulse input (2D delta function of frequencies w_1, w_2 as input x, i.e. $x_k(t) = \delta(\cos(w_k t) - 1), k \in \{1, 2\}$) and trained the network to generate a two cosines of the same frequencies, $y_k(t) = \cos(w_k t)$.

For the nonlinear RNN we designed a two frequency mixing task. Given two source frequencies, the network must generate two cosine waves whose frequencies are the sum and difference of two input frequencies. The network requires non-linearities in order to approximate the multiplication: $\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b), \cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$. In the task we provide the RNN with 2D periodic delta pulse of frequency $\frac{w1+w2}{2}$ and $\frac{w1-w2}{2}$ and learns to generate a trajectory of $\cos(w_1)$ and $\cos(w_2)$.

We use the following recurrent neural network,

$$\mathbf{g}(t+1) = f(\mathbf{W}_{\text{rec}}\mathbf{g}(t) + \mathbf{W}_{\text{in}} + \mathbf{b}_{\text{rec}}) \tag{6.41}$$

$$\mathbf{y}(t) = \mathbf{R}\mathbf{g}(t) + \mathbf{b}_{\text{out}} \tag{6.42}$$

in our linear RNNs $f(\cdot)$ is identity whereas in the nonlinear ones we used ReLU activation to enforce positivity condition.

For irrational output frequency ratio case, we used

$$w_1 = p\pi, w_2 = \sqrt{q}, p \sim \mathcal{U}(0.5, 4), q \ \sim \mathcal{U}(1, 10). \tag{6.43}$$

For rational case, we sampled

$$w_1, w_2 \in [1, 20] \cap \mathbb{Z}. \tag{6.44}$$

Where $\mathbb{Z}$ is the set of integers. For harmonics:

$$w_1 \in [1, 10] \cap \mathbb{Z}, w_2 = 2w_1. \tag{6.45}$$

We trained the RNN with the trajectory of length $T = 200$, bin size 0.1, and hidden dimension 16. For the linear RNN, we used learning rate 1e-3, 30k training iterations and $\lambda_{\text{target}} = 1$, $\lambda_{\text{activity}} = 0.5, \lambda_{\text{positivity}} = 5$, and $\lambda_{\text{weight}} = 0.02$. For the nonlinear RNN, we used learning rate 7.5e-4, 40k training iterations and $\lambda_{\text{target}} = 5, \lambda_{\text{activity}} = 0.5$ and $\lambda_{\text{weight}} = 0.01$. In both case, we initialised the weights to be orthogonal and the biases at zero, and used the Adam optimiser.

To assess the modularity of the trained RNNs, we performed a Fast Fourier Transform(FFT) on each neuron's activity and measured the relative power of the key frequencies w_1, w_2 with respect to the sum of total power spectrum

$$C_{\text{neuron}_i, w_j} = \frac{|FFT(g_i; w_j)|}{\sum_f |FFT(g_i; f)|} \tag{6.46}$$

and used it as a proxy of mutual information for modularity metric introduced in the coniditional mutual information metrics discussed by William Dorrell, K. Hsu, et al. (2025).

6.B.2 Nonlinear Teacher-Student RNNs

Network details. The Teacher network has input, hidden, and output dimensions of 2, and has orthogonal recurrent weights, input weights, and output weights. The Student RNN has input dimension 2, output dimension 2, and hidden dimension 64. It is initialised as per PyTorch default settings. The Teacher network dynamics is a vanilla RNN: $\boldsymbol{h}_t = tanh(\boldsymbol{W}_{rec}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{in}\boldsymbol{i}_t)$, and each teacher predicts a target via $\boldsymbol{o}_t = \boldsymbol{W}_{out}\boldsymbol{h}_t$. The Student RNN has identical dynamics (but with a ReLU activation function, and different weight matrices etc).

Generating training data. The teacher RNN generates training data for the Student RNN. We want to tightly control the Teacher RNN hidden distribution (for corner cutting or correlation analyses), i.e., tightly control the source distribution for the training data. To control the distribution of hidden activities of the Teacher RNN, we use the following procedure. 1) We sample a randomly initialised Teacher RNN. 2) $\boldsymbol{h}_0$ is initialised as a vector of zeros. 3) With a batch size of N, we take a **single** step of the Teacher RNN (starting from 0 hidden state) assuming $\boldsymbol{i}_t = 0$. This produces network activations (for each batch), $\boldsymbol{p}_t = tanh(\boldsymbol{W}_{rec}\boldsymbol{h}_{t-1}$. 4) We then sample from idealised distribution of the teacher hidden states, $\boldsymbol{h}_t$. For example a uniform distribution, or a cornet cut distribution. At this point these are just i.i.d. random variables, and not recurrently connected. 5) To recurrently connect these points, we optimise the input to the RNN, $\boldsymbol{i}_t$, such that the RNN prediction, $\boldsymbol{p}_t$, becomes, $\boldsymbol{h}_t$. We then repeat steps 3-5) for all subsequent time-steps, i.e., we find what the appropriate inputs are to produce hidden states as if they were sampled from an idealised distribution. To prevent the Teacher RNN from being input driven, on step 4), we solve a linear sum assignment problem across all batches (N

batches), so the RNN (on average) gets connected to a sample, $\boldsymbol{h}_t$, that is close to its initial prediction, $\boldsymbol{p}_t$. This means the input, $\boldsymbol{i}_t$, will be as small as possible and thus the Teacher RNN dynamics are as unconstrained as possible.

Training. We train the Student RNN on 10000 sequences generated by the Teacher RNN. The learning objective is a prediction loss $|\boldsymbol{o}_t^{teacher} - \boldsymbol{o}_t^{student}|^2$ plus regularisation of the squared activity of each neuron as well as each synapse (both regularisation values of 0.1). We train for 60000 gradient updates, with a batch size of 128. We use the Adam optimiser with learning rate 0.002.

Chapter 7

Neural Modules & Mixing: Cell Types and Mixed Grid Codes

During their explorations of behaving brains, neuroscientists have long categorised neurons by correlating their activities with some external variable. This has produced some of the greatest hits of neuroscience: orientation tuning curves (Hubel and Wiesel, 1962), place cells (O'Keefe and Dostrovsky, 1971), and grid cells (Hafting et al., 2005) to name a few. But why should we ever expect a neuron deep in the brain to encode meaningful variables (Hardcastle et al., 2017; Richards et al., 2019)? Here we use our biological constraints to normatively justify the existence of functional cell types, focusing on the plethora discovered in spatial neuroscience. These ideas additionally explain why the same cell might sometimes purely code one variable, and at other times mix two encodings: it depends on whether the two variables obey the modularising conditions in each task. Finally, we use our results to demonstrate how cells that are actually modular can appear mixed using common correlational analyses, potentially biasing our current understanding. In sum, we show how natural biological considerations encourage the emergence of meaningful modular structure, and how the same mathematical analyses can help us understand when and how this modularity will be broken.

Our theory of modularisation aims to be a useful guide for the structuring of neural representations across tasks and brain areas. As an example we study spatial processing. We show our biological constraints lead to separate neural populations (modules) coding for separate task variables, but **only** when task variables correspond to sufficiently independent factors of variation. Importantly, the modules consist of distinct functional cell types with similar firing properties, resembling grid (Hafting et al., 2005) and object-vector cells (Høydal et al., 2019).

We choose to focus on spatial representations for two reasons. First, there is a significant puzzle about why neurons deep in the brain, synaptically far from the sensorimotor periphery, almost miraculously develop single cell representations for human-interpretable factors (e.g. grid cells for location in space and object-vector cells for relative location to objects fig. 7.1C). Such observations are not easily accounted for by standard neural network accounts that argue that representations are unlikely to be human-interpretable (Richards et al., 2019). Secondly, whilst these bespoke spatial representations are commonly observed to factorise into single cells, there are situations in which selectivity spans across multiple task variables (Boccara et al., 2019; Butler, Hardcastle, and Giocomo, 2019; Hardcastle et al., 2017). Even more puzzlingly, these results appear context dependent. In different experiments grid cells appear to switch from pure spatial coding (H. Stensola et al., 2012), to a mixture of space and reward (Boccara et al., 2019; Butler, Hardcastle, and Giocomo, 2019). There is no existing theory to explain these puzzling patterns. We now partially plug this hole: the relationship bewteen reward and space differ in the experiments, when the relationship satisfies range-independence the representation is modular, in other settings they mix.

7.1 Range Independent Variables Produce Functional Cell Types

We demonstrate a task in which range-independent spatial variables naturally leads to factorised cell types, and use the same task to understand recent results showing the mixing of grid codes with rewards.

7.1.1 A Spatial Task and its Measured Cellular Responses

Spatial neurons are recorded as animals navigate environments, often scavenging for scattered rewards. In tasks such as these the spatial representation in rodent entorhinal cortex has been found to contain two distinct

modules of non-overlapping cell populations: (1) grid cells (Hafting et al., 2005) which represent allocentric space via hexagonal firing patterns; and (2) object-vector cells (Høydal et al., 2019) which represent relative location to objects, they fire fields at specific relative vectors from the object. These cells are usually measured in tasks where the relationship between rewarded locations and space is independent; classically in grid cell experiments food is randomly scattered around the environment to encourage the animal to explore (H. Stensola et al., 2012).

However, two recent papers examined the influence of inhomogeneous rewarded locations on the grid cell code, finding differing effects on the modularity of grid cells. Butler, Hardcastle, and Giocomo (2019) find that rewards rotate and cause fluctuations in the peak sizes of grid cells, but preserve their pure grid-like spatial firing pattern, while Boccara et al. (2019) find that the grid cells warp towards the rewards, becoming mixed-selective to reward-position and space (cells shown in fig. 7.1C). Crucially, however, the rewarding scheme differ between these two experiments: Boccara et al. (2019) fix the positions of the possible rewards during one day, whereas Butler, Hardcastle, and Giocomo (2019) alternate the animals between periods of free-foraging for randomly placed rewards and periods of specific rewarded locations. As such, in Butler, Hardcastle, and Giocomo (2019), since there are periods of randomly scattered rewards, reward-position and self-position are correlated but support-independent.

To encapsulate all these effects in one model, we consider a task in which rodents must know both where they are in space, and the action they would have to take in order to reach meaningful objects (i.e. rewards) in the environment. Across different environments the objects move around. If objects appear in different places in different contexts, the task is factorised into independent factors: 'Where am I in allocentric spatial coordinates?' and 'Where am I in object-centric coordinates?'. By contrast, if objects always appear in the same locations, the task is not factorised (as spatial location can predict reward location). We consider a parametrised family of tasks: at one extreme the objects always appear in two locations (red dots, fig. 7.1B), as in Boccara et al. (2019); at the other they appear in all positions with equal probability, as is done classically, e.g. H. Stensola et al. (2012); and intermediate tasks are made from these two extremes in different proportions, as in Butler, Hardcastle, and Giocomo (2019).

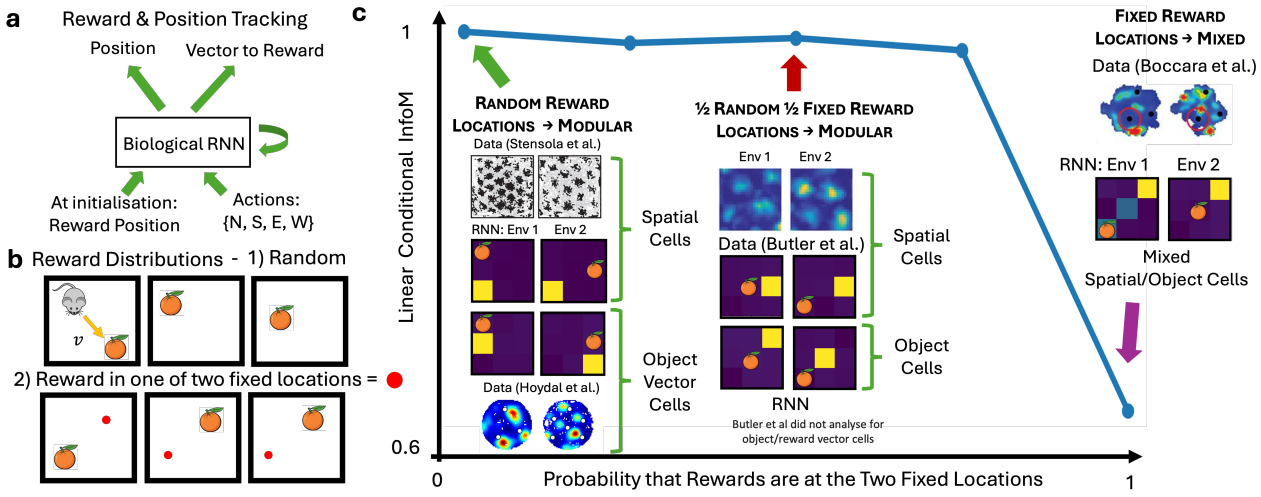


Figure 7.1: a, b) We train linear RNNs to integrate a sequence of actions and report their current position and displacement from a reward. In some proportion of the rooms, rewards are randomly scattered; in the rest, they are in one of two fixed locations. c) Matching the theory, if some of the rooms are random then reward and position are range-independent, and their representation modularises, as in H. Stensola et al. (2012) or Butler, Hardcastle, and Giocomo (2019); if in all rooms the rewards occur in one of two fixed locations the representation mixes, with neurons encoding both reward and self position, as in Boccara et al. (2019).

7.1.2 Biological RNN Model

We build a simple switching-linear RNN, much like the ones we used to model grid cells in part I. The internal dynamics of the RNN are action-dependent, allowing it to path-integrate:

$$\boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{W}_{\boldsymbol{a}}\boldsymbol{z}(\boldsymbol{x} - \boldsymbol{a}).$$

Further, we linearly readout two one-hot codes, one of the animal's position, another of its relative position from the reward. To match our biological constraints we enforce positivity in the neural activity and penalise an

energy efficiency loss made from weight and activity norms (details: section 7.A). This loss and architecture matches that studied in chapter 6, generalised to the case of multiple actions. Crucially, as discussed extensively in chapter 3, this loss will not create grid cells but place cells: specifically, it lacks the large number of positions and nonlinear decoding. Since it illustrates all the same points about mixed selectivity and is significantly simpler we make-do with place cells. Further details of the networks and tasks can be found in section 7.A.

7.1.3 Network Behaviour Matches Theoretical Predictions

Just as our recurrent theory predicts, chapter 6, when training on tasks where objects and space are factorised (i.e. objects can be anywhere in space), under our biological constraints of nonnegativity and energy efficiency, distinct neural modules emerge, each selective for a single task factor (Fig. 7.1C). We see spatial neurons that encode position independent of object locations, and object-vector-like neurons that recentre their representations around the moving objects. Further, the two populations remain modular if there is any probability that the rewards will appear in every point in the room. It is only in the extreme case, when the rewards appear in only two points in the room, that the cells become mixed selective, as in Boccara et al. (2019).

7.2 Missed sources and mixed selectivity

In contrast to the beautifully modular neurons in entorhinal cortex, recent work has highlighted neurons with mixed tuning to multiple navigational variables, such as position, heading direction, or speed (Hardcastle et al., 2017). We use our theory to highlight a potential artifact of this type of analysis. A purportedly mixed selective neuron may (in part) be purely selective for another unanalysed variable that is itself correlated with the measured spatial variables or behaviour. We highlight a simple example of this effect in fig. 7.2. Imagine a mouse is keeping track of three objects as it moves in an environment (fig. 7.2a); we model this as a linear RNN with biological constraints that must report the displacement of all objects from itself. We make the object positions range independent but correlated. As predicted by our theory, the RNN modularises these range-independent variables such that each neuron only encodes one object (fig. 7.2c). However, if an experimenter were only aware of two of the three objects, they would instead analyse the neural tuning with respect to those two objects, without being able to condition on the third. Due to the statistical dependencies between object positions, they would find mixed selective neurons that are in reality purely coding for the missing object (fig. 7.2d). We do not believe that all neurons in entorhinal cortex are modular; for example, there is clear evidence that neurons co-tuned to position and velocity are meaningfully mixed-selective, implementing the path-integration updates in the grid cell system (Vollan et al., 2025). Indeed, many of the neurons analysed in Hardcastle et al., 2017 *likely are* mixed selective. However, it is also seems likely that in more exploratory analyses such as these, mixed selectivity might often arise from missing encoded variables in the analysis.

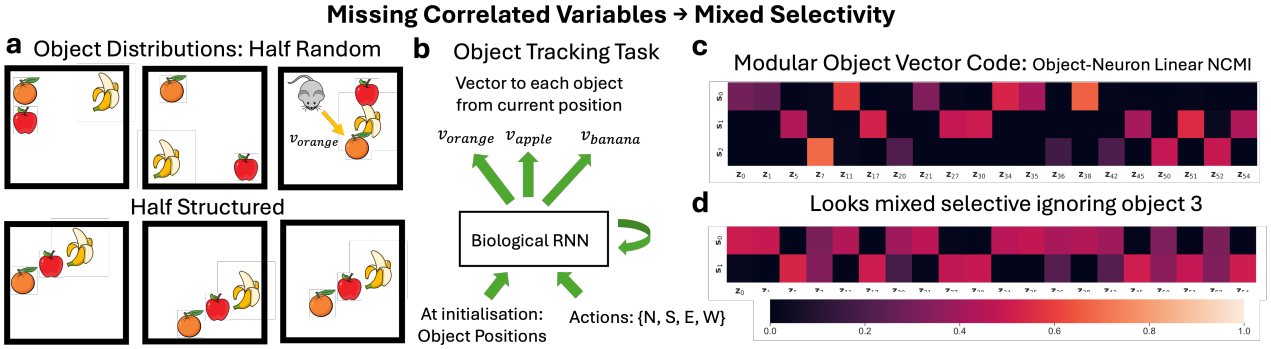


Figure 7.2: a, b) We train linear RNNs to report displacement to three objects as an agent moves within many rooms. In each room, object instances are either completely random (a, top) or clustered in a random line (a, bottom); object positions are therefore range independent but correlated. c) The neurons are modular: each neuron's activity is conditionally informative of only a single object given the others. d) However, if an experimenter were only aware of two of the three objects, the neurons that purely encode the disregarded object would appear mixed selective due to the statistical dependencies between objects.

Technical Appendicies: Neural Modules and Mixing

7.A RNN Models of Entorhinal Cortex

We now talk through the linear RNNs used in this chapter. Inspired by our work on grid cells. In each trial the agent navigates a 3x3 periodic environment. The RNN receives one special input only at timepoint 0 that tells it the layout of the room (format described later). Otherwise at each timestep it is told which action (north, south, east, west), the agent took. It uses this action to linearly update its hidden state via an action-dependent affine transform:

$$\boldsymbol{g}_t = \boldsymbol{W}_{a_t}\boldsymbol{g}_{t-1} + \boldsymbol{b}_{a_t} \tag{7.1}$$

At each timestep it has to output, via an affine readout, a particular target depending on the task.

In the spatial-reward task, fig. 7.1, at timepoint 0 the agent starts at a consistent position and receives an input that is a 9-dimensional 1-hot code telling the agent the relative position of the reward. At each timestep the agent then has to output two 9-dimensional 1-hot codes, one signalling its position in the room, the other the relative displacement of the reward from its position.

The agent experiences many rooms. Across rooms the rewards are either randomly sampled (random rooms), or sampled from one of two fixed positions (fixed rooms). Each agent experiences a different mixture of randomly sampled or fixed rooms. We then optimise the network to perform the task with non-negative neural activities while penalising the L2 norm of the activity and all weight matrices.

In the mixed-selectivity task, fig. 7.2, there are three objects in each room and the agent has to report the relative displacement of each of them. The input at the start of each trial is a 27-dimensional code signalling where the three objects are relative to its own position. As such, as the agent moves around the room it has to keep track of the three objects. This task was harder to train so we moved from the mean squared error to the cross-entropy loss and found it worked well, apart from that all details of loss and training are the same.

Between trials we randomise the position of the objects. Some portion (0.8) of the time these positions are drawn randomly (including objects landing in the same position). The rest of the time the objects were positioned so that the first object was one step north-east of the second, which itself was one step north-east of the third. This introduced correlations between the positions of the objects, while preserving their range independence - all objects could occur in all combinations.

We measured the linear NCMI between the neural activity and the 27-dimensional output code and found that each neuron was informative about a single source, as expected, it had modularised fig. 7.2C. However, pretend we did not know the third object existed. Instead we would calculate the linear NCMI between each neuron and those objects we know to exist. We would still find modular codes for these objects, but we would also find that the neurons that in reality code for third object, due to the correlations between object placements, are actually informative about the first and second object, so they look mixed selective! The position of the objects is always informative about each other, but by conditioning on each object we are able to remove this effect with our metric and uncover the latent modularity. But without knowing which latents to condition on we cannot proceed, and instead get lost in correlations.

Chapter 8

Convex Efficient Coding

Why do neurons encode information the way they do? Normative approaches are a common and satisfying answer to this question, for example, the efficient coding hypothesis models neural activity as the optimal encoding of some information under efficiency constraints. Successful examples have varied dramatically in complexity, from simple sparse linear models (Olshausen and Field, 1996), to complex regularised deep neural networks (Lindsay, 2021). While complex models are flexible, they lack tractability, limiting their scientific value. Here, we partially bridge this gap by constructing a set of tractable but flexible normative representational theories. Instead of optimising the neural activities directly, we build on Sengupta et al. (2018) and rewrite the problems as optimisations over the representational similarity, a matrix whose entries are the dot products of each neural encoding $\boldsymbol{Q}_{t,t'} = \boldsymbol{z}_t^T \boldsymbol{z}_{t'}$. Using this, we show that a large family of interesting problems are convex. This includes both problems that are effectively linear and non-linear neural networks, and problems studied in the literature but not previously recognised as convex.

We then put these findings to work. First, we extend previous results on modularity and mixed selectivity in neural activity; in so doing we provide the first necessary and sufficient identifiability result for matrix factorisation. Second, we wonder how robustly tuning curves can be used to infer the function of a neural system by asking the question 'given a set of tuning curves when does every optimal representation to contain the same responses?' We derive a similar identifiability result: if neural tunings are 'different enough', then they are robustly indicative of function, justifying their use in neuroscience. Finally, we turn to a neural coding question: why does the retina split information into ON and OFF channels but not V1? Using our tractable nonlinear models we develop a normative theory that emphasises the role sparsity: a dense code, like in retina, optimally uses OFF and ON channels, while a sparse code, like in V1, does not. In sum, we open a space of problems to tools from convex analysis, and use this to derive results of interest to both neuroscience and machine learning.

8.1 Introduction

Neural activity forms an encoding of information, both in nervous systems and artificial neural networks. Towards the broader goal of understanding these systems, a lot of work has studied the properties of these encodings, for example, is neural activity a temporal or a rate code (Gautrais and Thorpe, 1998), or why do we find Gabor filters in both visual cortex (Jones and Palmer, 1987) and Convolutional Neural Networks trained to recognise objects (Yosinski et al., 2014). Normative approaches provide compelling answers to these questions, the most classical of which is the efficient coding hypothesis (Attneave, 1954; Barlow et al., 1961); loosely - 'neurons use the representation that most efficiently encodes the required information'. Finding a match between optimal encodings and the brain provides evidence that these neurons can be thought of as solving the proposed optimisation problem, naturally leading to predictions and broader functional theories. These approaches have been widely successful, for example in studying tuning curves (Laughlin, 1981), including the gabor filters cited above (Okajima, 1998a,b; Olshausen and Field, 1996).

However, the tractability of these theories varies significantly. Some problems admit pleasing analytic solutions which lay bare all aspects of the problem's structure (Atick and Redlich, 1992). More often this is not the case. For example, often neural activity is modelled using a task-optimised neural network (Botvinick and Plaut, 2006; Gauthier and Levy, 2019; Goldstein et al., 2022; G. Hinton et al., 2006; Kell et al., 2018; Lindsay, 2021; Tanaka, 2016; J. X. Wang et al., 2018; Yamins et al., 2014; Zipser and Andersen, 1988). Currently, these approaches are, for the most part, not analytically tractable, especially when the models are large, hindering the insights that could be gained.

A lot of work therefore attempts to 'open the black box', either by directly studying trained networks, or using analytic models. In this work we build on a recent example of a tractable neural optimisation problem. Sengupta et al. (2018) study a similarity matching objective and show that the optimal nonnegative representation of compact spaces is place cells if there are unlimited neurons. The technical novelty of this approach lies in showing that the optimisation problem can be written as a convex optimisation problem over the set of representational dot-product similarity matrices. In so doing, Sengupta et al. (2018) unlock the application of convex analysis to cutting-edge problems in neuroscience. However, this approach has not been extensively used, perhaps due to the bespoke nature of the similarity matching objective.

Here, we broaden these approaches. In section 8.2, we show that we can write a large family of objectives and constraints as convex over the set of representational dot-product similarity matrices. Arbitrary sets of convex constraints and objectives can then be composed to create a variety of interesting optimisation problems, each of which are convex. Many of these problems correspond to linear, or even nonlinear, neural networks with different regularisation schemes. Indeed, using these components, some problems from the literature can be reframed as a convex problem. It seems likely that other interesting representational phenomena can be framed in this way and composed with the results presented here.

We then use these results in three ways. First, in section 8.3, we study a problem from the literature that was not previously identified as convex: the positive-linear autoencoding of a set of sources (variables) (William Dorrell, K. Hsu, et al., 2025; J. C. Whittington, Will Dorrell, et al., 2023). Previous works studied orthogonally embedded sources and derived a set of necessary and sufficient conditions such that optimally each latent neuron encodes a single source, and used this to model patterns of modularity and mixed-selectivity in the entorhinal cortex. Here, we generalise these results to the case of linearly (rather than orthogonally) mixed sources. This change seems small, but in so doing, we derive the first necessary and sufficient conditions for the identifiability of matrix factorisation. These identifiability results describes the conditions under which your model will correctly extract the data's structure - in effect, summarising the model's assumptions about the world.

Second, in section 8.4, we turn to single-neuron tuning curves. Normative approaches assume that the measured neural responses reflect some kind of optimal solution. A pertinent question is therefore how loose the class of optimal solutions is for a given problem. For example, if the encoding can be rotated arbitrarily, scrambling the neural tuning curves without changing performance, then looking at tuning curves seems short-sighted. In order for the neuron-axes to be meaningful something must break the rotational symmetry of the problem. One of our convex constraints is that neural activity must be nonnegative, breaking this symmetry. For problems with this constraint, we derive a set of sufficient conditions that outline how, if the tuning curves are 'different enough', all optimal solutions will contain the same neural responses. In so doing, we provide a justification for studying tuning curves, linking them precisely to the representation's computational goal.

Finally, in section 8.5, we consider an existing neural coding question: why do retinal cells split into ON and OFF channels, producing pairs of neurons encoding oppositely rectified versions of the same signal, but V1 doesn't? Previous work has shown empirically that ON-OFF coding schemes emerge in task-optimised neural networks (Ocko et al., 2018), or shown that often an ON-OFF code is more efficient (Gjorgjieva, Sompolinsky, and Meister, 2014), but they lacked a tractable normative framework to explain this choice. This is because ON-OFF coding is inherently nonlinear, it's simply a pair of rectifications, stalling standard linear approaches. We make use of the nonlinear tractability of our problems to show that this coding choice is governed by sparsity: dense codes, like retina, are optimally split into two channels, while sparse codes, like V1, are not.

In sum, we provide a theoretical framework based on convex analysis that provides tractability for problems in neuroscience and machine learning, and we use this unified view to derive solutions and guarantees for long-standing questions in matrix factorisation and neural coding.

8.2 A Family of Convex Representational Optimisations

We begin by establishing the convexity of a series of interesting representational optimisation problems.

Motivating Example We consider optimisation problems over representations: d_z-dimensional neural encodings of a set of T datapoints: $\boldsymbol{z}_t \in \mathbb{R}^{d_z}$. For example, let's imagine a set of targets $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$, and require them to be linearly decoded from the representation:

$$\exists \boldsymbol{W}_{\text{out}} \in \mathbb{R}^{d_y \times d_z} \text{ such that:} \qquad \boldsymbol{W}_{\text{out}} \boldsymbol{z}_t = \boldsymbol{y}_t \quad \forall t = 1, ..., T \tag{8.1}$$

We might additionally constrain the neural activity to be nonnegative ($\boldsymbol{z}_t \geq 0$), and then penalise the energy use of the representation, either through firing rates or synaptic weights, in line with evidence that the largest energy cost of spiking is synaptic transmission (J. J. Harris, Jolivet, and Attwell, 2012). Hence our problem:

$$\min \langle ||\boldsymbol{z}_t||^2 \rangle_t + \lambda ||\boldsymbol{W}_{\text{out}}||_F^2 \quad \text{subject to} \quad \boldsymbol{z}_t \geq 0, \quad \boldsymbol{W}_{\text{out}} \boldsymbol{z}_t = \boldsymbol{y}_t \tag{8.2}$$

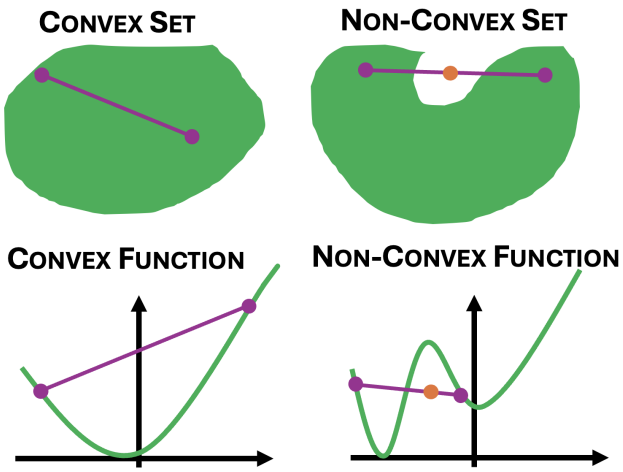


Figure 8.1: Illustrations of the definitions of a convex set and function. The top right set is non-convex because the orange point is on the line of interpolation between two members of the set (purple points) but is not itself in the set. No such examples exist in the top left. Similarly, in the lower right the orange point is on the interpolation of two function evaluations but is below the function evaluated at the interpolation, demonstrating non-convexity; no such examples exist in the case of the convex function.

This is a very simple instantiation of the efficient coding hypothesis - encode information subject to energy constraints - but where we assume the decoder is linear. Yet, even this simple model has widespread use in neuroscience (Dordek et al., 2016; Martín-Sánchez, Machens, and Podlaski, 2025; Sorscher, G. Mel, et al., 2019). In this work we show that problems like these can be framed as convex optimisations over the set of representational dot-product similarity matrices. The single largest limitation of our work is that, in order to make the problem convex, we have to assume the number of neurons is larger than the number of datapoints, $d_z \geq T$. Even if this is not the case then, as we'll discuss in section 8.6, all is not lost, but for now we take it for granted.

An optimisation problem such as $x^* = \min_{x \in \mathbb{X}} f(x)$ is convex if:

1. $\mathbb{X}$ is a convex set.

   This means that, if x and x' are both members of $\mathbb{X}$, then so is $\alpha x + (1-\alpha)x'$ when $0 \leq \alpha \leq 1$. Intuitively this makes sure the set isn't too 'spiky', fig. 8.1.

2. $f(x)$ is a convex function.

   This means that the function evaluated at an interpolation between two points is always lower than the interpolation of the function at those points: $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \forall 0 \leq \alpha \geq 1$. Intuitively this also constrains the 'spikiness' of the function, fig. 8.1.

Following Sengupta et al. (2018), we will rewrite the problem in terms of the representational dot-product similarity matrix, defined as $(\boldsymbol{Q})_{t,t'} = \boldsymbol{z}_t^T \boldsymbol{z}_{t'}$. It turns out the problem can be written entirely in terms of this matrix, yielding convexity in both the constraints and the objectives. We'll now partially illustrate this:

- *Firing Energy:* $\langle ||\boldsymbol{z}_t||^2 \rangle_t = \frac{1}{T} \sum_t \boldsymbol{z}_t^T \boldsymbol{z}_t = \frac{1}{T} \text{Tr}[\boldsymbol{Q}]$ which is convex since $\text{Tr}[\boldsymbol{Q}]$ is a linear function of $\boldsymbol{Q}$.
- *Weight Energy:* Since the representation is linearly decodable, the min-norm choice of readout weight matrix is given by the psueodinverse. Defining the data and representation matrices:

$$\boldsymbol{Y} \in \mathbb{R}^{d_y \times T}, [\boldsymbol{Y}]_{:,t} = \boldsymbol{y}_t \qquad \boldsymbol{Z} \in \mathbb{R}^{d_z \times T}, [\boldsymbol{Z}]_{:,t} = \boldsymbol{z}_t \qquad \boldsymbol{W}_{\text{out}} = \boldsymbol{Y}\boldsymbol{Z}^\dagger \tag{8.3}$$

  Then the loss becomes:

$$||\boldsymbol{W}_{\text{out}}||_F^2 = \text{Tr}[\boldsymbol{W}_{\text{out}}^T \boldsymbol{W}_{\text{out}}] = \text{Tr}[\boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{Z}^\dagger \boldsymbol{Z}^{\dagger,T}] = \text{Tr}[\boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{Q}^\dagger] \tag{8.4}$$

  Which we show in section 8.A is a convex function of $\boldsymbol{Q}$.

- *Nonnegativity:* The set of dot-product matrices of nonnegative vectors ($\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$, $\boldsymbol{Z} \geq 0$) form a convex set called the set of completely positive matrices (Berman and Shaked-Monderer, 2003), as in Sengupta et al. (2018).

We show in section 8.A that the final constraint - that the readouts are linearly decodable from the representation - is also convex, and hence that the problem convex. This ensures that all locally optimal $\boldsymbol{Q}$ are globally optimal, and opens the problem to tools from convex optimisation (though characterising the set of completely positive matrices is an unsolved problem (Dür, 2010), limiting the power of some tools, section 8.6).

A Family of Convex Problems This was a simple example problem. In section 8.A we similarly show that each of the following constraints and objectives are convex over the set of representational similarity matrices, from which we can construct a series of problems of interest.

Constraints

C1 Nonnegativity: $\boldsymbol{Z} \geq 0$ (As in Sengupta et al. (2018))

C2 Full rank affine relationship between representation and input dataset such that $\boldsymbol{Y}$ can be decoded, $\boldsymbol{X} \in \mathbb{R}^{d_x \times T}$: $\boldsymbol{W}_{\text{in}}\boldsymbol{X} + \boldsymbol{b}_{\text{in}}\boldsymbol{1}^T = \boldsymbol{Z}$, $\text{rank}(\boldsymbol{W}_{\text{in}}) = d_x$. (Can be relaxed to linear)

C3 Perfect affine decodability of a dataset, $\boldsymbol{Y} \in \mathbb{R}^{d_y \times T}$: $\boldsymbol{W}_{\text{out}}\boldsymbol{Z} + \boldsymbol{b}_{\text{out}}\boldsymbol{1}^T = \boldsymbol{Y}$. (Can be relaxed to linear)

C4 A ReLU relationship between representation and input dataset, $\boldsymbol{X} \in \mathbb{R}^{d_x \times T}$: $\boldsymbol{Z} = \text{ReLU}(\boldsymbol{W}_{\text{in}}\boldsymbol{X} + \boldsymbol{b}_{\text{in}}\boldsymbol{1}^T)$

C5 A firing rate constraint: $||\boldsymbol{z}_i||_2^2 \leq k \forall i$. (As in Sengupta et al. (2018))

Objectives

O1 L2 activity loss: $||\boldsymbol{Z}||_F^2$

O2 A modified L1 activity loss: $\sum_d ||\boldsymbol{z}_d||_1^2$, when the representation is nonnegative (C1)

O3 An L2 input weight loss when the relationship with input data is affine (C3): $||\boldsymbol{W}_{\text{in}}||_F^2$

O4 An L2 output weight loss when the data is perfectly affine decodable (C3): $||\boldsymbol{W}_{\text{out}}||_F^2$

O5 Similarity matching for a dataset $\boldsymbol{X}$, as in (Sengupta et al., 2018): $-\text{Tr}[(\boldsymbol{X}^T\boldsymbol{X} - \alpha\boldsymbol{1}\boldsymbol{1}^T)\boldsymbol{Z}^T\boldsymbol{Z}]$.

O6 Nonlinear similarity matching with input similarity $\chi \in \mathbb{R}^{T \times T}$ and elementwise exponentiation: $\text{Tr}[\chi e^{\boldsymbol{Z}^T\boldsymbol{Z}}]$.

Convex Problems

- A regularised, perfectly fitting, linear neural network as in A. M. Saxe, McClelland, and Ganguli (2019).
- A regularised, perfectly fitting, one-layer ReLU network.
- A positive affine autoencoder, as in William Dorrell, K. Hsu, et al. (2025) and J. C. Whittington, Will Dorrell, et al. (2023).
- A nonlinear but linearly-decodable representation such as the last layer of an unconstrained neural network.
- A representation that nonlinearly matches the data similarity, such as the latents in a nonlinear autoencoder.

We now turn to using this convexity in three settings: matrix factorisation learning, the unqueness of neural tuning curves, and channel splitting in neural codes.

8.3 Necessary & Sufficient Identifiability of Matrix Learning

Background Matrix factorisation problems seek to break a data matrix, $\boldsymbol{Y} \in \mathbb{R}^{d_Y \times T}$, into two meaningful parts $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S}$, $\boldsymbol{A} \in \mathbb{R}^{d_Y \times d_s}, \boldsymbol{S} \in \mathbb{R}^{d_s \times T}$. For example, dictionary learning seeks learns a dictionary, $\boldsymbol{A}$, and sources, $\boldsymbol{S}$, that fit the data while using a sparse $\boldsymbol{S}$. In general, there will be many possible choices of $\boldsymbol{A}$ and $\boldsymbol{S}$ that recreate the same $\boldsymbol{Y}$. For example, given one feasible pair $(\boldsymbol{A}, \boldsymbol{S})$ such that $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S}$, inserting any invertible $d_s \times d_s$ matrix, $\boldsymbol{B}$, will generate another feasible pair, $(\boldsymbol{A}\boldsymbol{B}, \boldsymbol{B}^{-1}\boldsymbol{S})$, since: $\boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{S} = \boldsymbol{A}\boldsymbol{S} = \boldsymbol{Y}$. Therefore, useful problem settings try to find effectively unique decompositions either by constraining the allowed factorisations or regularising their choice. For example, semi-nonnegative matrix factorisation problems constrain one of the matrices to be nonnegative, usually $\boldsymbol{S}$, while sparse coding combines this with an L1 regularisation on the source matrix (Olshausen and Field, 1996). However, even when the factors are carefully constrained, the measured data still has to play ball. For example, if $\boldsymbol{Y} = \boldsymbol{0}$ then $\boldsymbol{A}$ can take any value as long as $\boldsymbol{S} = \boldsymbol{0}$, meaning that, for example, sparse coding is not well-posed.

Identifiability Identifiability results outline when a problem is well-posed. They assume the data-generating assumptions are true, for example that the data really is linearly generated from two sub-matrices (otherwise how do we know what the model 'should' do?), and outline under which conditions the proposed algorithm will recover the true data-generating factors. Previous work has extensively studied the identifiability of matrix factorisation, as we'll review later. However, almost all results are either necessary or sufficient, but not both. Here, we use our convexity results, section 8.A, to derive necessary and sufficient identifiability conditions for a neural network based factorisation algorithm with applications in theoretical neuroscience: positive affine autoencoders.

Problem 1 (Positive Affine Autoencoding)**.** *Let $\boldsymbol{s}^{[i]} \in \mathbb{R}^{d_s}$, $\boldsymbol{x}^{[i]} \in \mathbb{R}^{d_x}$, $\boldsymbol{z}^{[i]} \in \mathbb{R}^{d_z}$, $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{d_z \times d_x}$, $\boldsymbol{b}_{\text{in}} \in \mathbb{R}^{d_x}$, $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{d_x \times d_z}$, $\boldsymbol{b}_{\text{out}} \in \mathbb{R}^{d_x}$, and $\boldsymbol{A} \in \mathbb{R}^{d_x \times d_s}$ where $d_z > d_s$ and $d_x \geq d_s$, and $\boldsymbol{A}$ is rank d_s. We seek the solution to the following the constrained optimization problem.*

$$\begin{aligned} \min_{\boldsymbol{W}_{\text{in}}, \boldsymbol{b}_{\text{in}}, \boldsymbol{W}_{\text{out}}, \boldsymbol{b}_{\text{out}}} \quad & \left\langle ||\boldsymbol{z}^{[i]}||_2^2 \right\rangle_i + \lambda \left(||\boldsymbol{W}_{\text{in}}||_F^2 + ||\boldsymbol{W}_{\text{out}}||_F^2 \right) \\ \text{s.t.} \quad & \boldsymbol{z}^{[i]} = \boldsymbol{W}_{\text{in}} \boldsymbol{x}^{[i]} + \boldsymbol{b}_{\text{in}},\ \boldsymbol{x}^{[i]} = \boldsymbol{W}_{\text{out}} \boldsymbol{z}^{[i]} + \boldsymbol{b}_{\text{out}},\ \boldsymbol{z}^{[i]} \geq 0, \end{aligned} \tag{8.5}$$

where i indexes a finite set of samples of $\boldsymbol{s}$, and $\boldsymbol{x}^{[i]} = \boldsymbol{A}\boldsymbol{s}^{[i]}$.

This is a factorisation problem that, up to a constant shift, breaks the data into two matrices: a weight matrix and a neural representation, $\boldsymbol{Z}$. We look for identifiability conditions under which $\boldsymbol{Z} = \Pi \boldsymbol{S} + \boldsymbol{b}\boldsymbol{1}$, meaning the latents recover the sources up to an arbitrary offset $\boldsymbol{b}$, and a 'rectangular permutation' matrix with at most one non-zero entry in each row (and at least d_s non-zero entries). Previous work has derived conditions for the case when $\boldsymbol{A}$ is an orthogonal matrix, and used this to understand patterns of modularity and mixed-selectivity in the entorhinal cortex (William Dorrell, K. Hsu, et al., 2025; J. C. Whittington, Will Dorrell, et al., 2023). Here, we use the convexity of the optimisation over the set of representational similarity matrices, $\boldsymbol{Z}^T\boldsymbol{Z}$, section 8.2, to derive identifiability results for the case of arbitrary $\boldsymbol{A}$, dropping the orthogonality condition. In particular, we will show that it will be governed by a 'tight scattering' condition that characterises whether the sources are sufficiently scattered with respect to a particular $\boldsymbol{A}$-dependent ellipse:

Definition 8.3.1 (Tight Scattering)**.** *First, generate the de-meaned sources: $\bar{\boldsymbol{S}} = \boldsymbol{S} - \boldsymbol{S}\boldsymbol{1}\boldsymbol{1}^T \in \mathbb{R}^{d_S \times T}$ with elements $\bar{S}_{dt} = \bar{s}_d^{[t]}$. Assume without loss of generality that $|\min_t \bar{s}_d^{[t]}| \leq \max_t \bar{s}_d^{[t]}$ for each dimension d (if this is not satisfied simply redefine this source as $-\boldsymbol{s}_d$), then construct the diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{d_S \times d_S}$ and the symmetric matrix $\boldsymbol{F}^{d_S \times d_S}$:*

$$D_{jj} = \sqrt[4]{\frac{\lambda (\boldsymbol{A}^T\boldsymbol{A})_{jj}}{\langle (\bar{s}_j^{[i]})^2 \rangle_i + (\min_i \bar{s}_j^{[i]})^2 + \lambda ((\boldsymbol{A}^T\boldsymbol{A})^{-1})_{jj}}} \qquad \boldsymbol{F} = \lambda \boldsymbol{D}^{-2} (\boldsymbol{A}^T\boldsymbol{A}) \boldsymbol{D}^{-2} - \lambda (\boldsymbol{A}^T\boldsymbol{A})^{-1} - \bar{\boldsymbol{S}}\bar{\boldsymbol{S}}^T \tag{8.6}$$

Then use these matrices to construct the following set:

$$E = \{\boldsymbol{x} | \boldsymbol{x}^T \boldsymbol{F}^{-1} \boldsymbol{x} = 1\} \tag{8.7}$$

Then $\boldsymbol{S}$ is tightly scattered with respect to a generating matrix $\boldsymbol{A}$ if the following conditions hold:

- *Conv$(\bar{\boldsymbol{S}}) \supseteq E$*
- *Conv$(\bar{\boldsymbol{S}})^* \cap bdE^* = \{\lambda \boldsymbol{e}_k, \lambda \neq 0 \in \mathbb{R}, k = 1, ..., d_S\}$ where the $*$ denotes the dual cone, and bd the boundary.*

The first condition ensures that the convex hull of the sources engulfs the ellipse. The second is a technical condition that ensures the ellipse and the convex hull of the sources only touch along basis directions. The following theorem simply states that if the sources are tightly scattered the problem is identifiable.

Theorem 5 (Identifiability of Positive Affine Autoencoders)**.** *Given a dataset $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$, if the matrix $\bar{\boldsymbol{S}}$ is tightly scattered with respect to $\boldsymbol{A}$ then the optimal positive affine autoencoder recovers the sources - each neuron's activity is an affine function of one source and every source has at least one neuron encoding it:*

$$\boldsymbol{z}_n^{[i]} = d_n (s_n^{[i]} - \min_i [s_n^i]) \tag{8.8}$$

Proof. See section 8.3. □

In fig. 8.2 we show a simple example, we create a linear mixture of two sources. We then plot the set E for two values of the regularisation hyperparameter λ, and see that it changes whether or not the set E is a subset of the convex hull of the sources, satisfying our condition. We train the optimal affine autoencoder and see that our condition correctly predicts which optimal representation is modular.

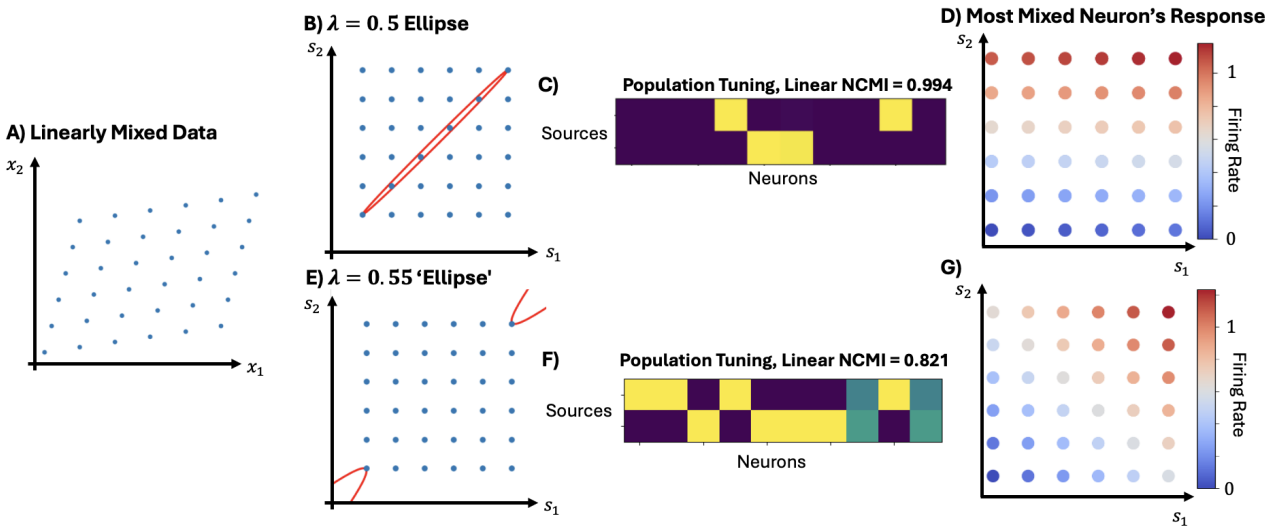


Figure 8.2: **(A)** We linearly mix sources with a matrix $\boldsymbol{A}$ and consider the optimal representation for two values of λ. In one case **(B)** the matrix $\boldsymbol{F}$ is positive definite, meaning E is an ellipse, and we see that, in this case, the convex hull of the data encloses E. Thus, theory predicts that the representation should modularise. **(C)** The empirics agree, we calculate the conditional linear mutual information (William Dorrell, K. Hsu, et al., 2025; K. Hsu, William Dorrell, et al., 2023) between each neuron and the sources and find that each neuron is tuned to a single source. Further, **(D)** we visualise the response of the 'most mixed' neuron and find that it is clearly modular. Yet **(E-G)** slightly increasing λ makes the matrix $\boldsymbol{F}$ indefinite, E becomes a parabola that no finite dataset can enclose in its convex hull, and we get a mixed optimal representation, as shown to the right.

Previous Work We conclude this section by relating to previous work. The identifiability of various matrix factorisation problems has been extensively studied. Donoho and Stodden (2003) studied nonnegative matrix factorisation and showed that if the data satisfied a scattering condition (that the convex cone of datapoints surrounded an 'ice-cream cone' in the postive orthant) the model was identifiable. For dictionary learning a lot of work has shown that a hard sparsity constraint on the sources is sufficient (Aharon, Elad, and Bruckstein, 2006; J. E. Cohen and Gillis, 2019; Hillar and Sommer, 2015), though the required sample size for these bounds to work can be prohibitively large. J. Hu and Huang (2023) present a state-of-the-art sufficient identifiability condition regarding the scattering of a sign-permuted dataset around a unit ball. This is similar both to our work, to work on polytope matrix factorisation (Tatli and Erdogan, 2021a,b) and to sufficient spread conditions for identifiabilty of nonnegative matrix factorisation (Huang, Sidiropoulos, and Swami, 2013). Our work differs in that it adapts the identifiability condition depending on the mixing matrix, $\boldsymbol{A}$, allowing us to derive a necessary and sufficient condition for identifiability, to our knowledge the first such condition derived.

8.4 When can I infer function from my Tuning Curves?

A classic approach in neuroscience is to measure neural tuning curves and use their properties to reason about the underlying computations. For example, centre-surround receptive fields in early visual processing can be understood as performing whitening (Atick and Redlich, 1990, 1992). This is only reasonable if there is some link between the tuning and computation, a natural assumption. However, a recent paper has highlighted that many neural networks can perform the same computation while using very different representations (Braun, E. Grant, and A. M. Saxe, 2025) (in their case, they analytically study linear networks). Hence, they find that function and representation are 'fundamentally disassociated' prompting questions about the legitimacy of classical approach in neuroscience.

However, Braun, E. Grant, and A. M. Saxe (2025) also point out that the linear network that solves a problem with the minimum weight norm has a unique representational dot-product similarity. Indeed, the networks they study fall into the class of convex optimisation problems discussed above, section 8.2, so it makes sense that they have a globally optimal dot-product similarity structure. This instead suggests that, when we study the optima of reasonably posed problems, representation and function are in fact tightly coupled, making the classic neuroscientific approach a reasonable one. As such, in all the convex problems listed in section 8.2 the coupling between function and representational similarity is similarly tight, thanks to convexity. Techniques like RSA (Kriegeskorte, Mur, and Bandettini, 2008), which use the representational similarity matrix, are therefore well-suited to studying representations that solve the problems introduced in section 8.2.

We now take this argument one step further. RSA is most often used for broadscale measurement methods, like fMRI. Neural data, on the other hand, provides microscopic information on the behaviours of single neurons. These single neurons often appear meaningful beyond just contributing to the population structure, such as grid cells (Hafting et al., 2005) which seem clearly linked to path-integration (McNaughton et al., 2006). When should we expect single neuron tuning to be as tightly coupled to function as the representational similarities are in our family of convex problems?

Here, we present a neural tuning curve identifiability result. If the neural encoding can be rotated without effecting computation then clearly the single neuron responses won't be uniquely coupled to the optimal computation, all rotations will also suffice, and these can arbitrarily scramble the neural responses. However, if the neural encoding is nonnegative, arbitrary rotations will no longer be allowed. We therefore study strictly convex optimisation problems over the set of representational similarity matrices subject to a nonnegativity constraint on neural firing. We find that if the single neuron tuning curves are different enough, we can guarantee that all optimal solutions will contain these neurons, creating a tight coupling between sets of single neuron tuning curves and their function.

Precise Problem Statement and Sufficient Conditions Let's say you measure a nonnegative neural representation, $\boldsymbol{Z}^*$, which you assume achieves some optimal similarity structure: $\boldsymbol{Z}^{*,T}\boldsymbol{Z}^* = \boldsymbol{Q}^*$. What has to be true about $\boldsymbol{Z}^*$ such that all other optimal representations contain the same neural tuning curves?

In order to be optimal a putative representation must have the same dot-product similarity: $\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{Q}^\star$. Therefore, it must be an orthogonal transform of the first representation: $\boldsymbol{Z} = \boldsymbol{O}\boldsymbol{Z}^*$, while still preserving nonnegativity. We present results that constrain $\boldsymbol{O}$ to being a permutation matrix, meaning that every optimal representation contains the same neural tuning curves. In particular, this is true if the neural response patterns, or tuning curves, satisfy a family of sufficient scattering conditions, closely related to those in section 8.3. In effect, this says that if your neural responses are 'different enough', in some precise sense, then all optimal models will possess these patterns. This is described in the following theorem, proofs are presented in section 8.C.

Theorem 6 (Tight Scattering Implies Unique Neurons)**.** *If your neural data, $\boldsymbol{z}^{[i]}$, satisfies the following two tight scattering constraints with respect to a set $E = \{\boldsymbol{x} + \langle \boldsymbol{z}^{[i]}\rangle_i | \boldsymbol{x}^T\boldsymbol{F}^{-1}\boldsymbol{x} = 1\}$ for a positive definite matrix $\boldsymbol{F}$ with diagonals equal to $(\langle \boldsymbol{z}^{[i]}\rangle_i)^2$ then all orthogonal matrices such that $\boldsymbol{O}\boldsymbol{z}^{[i]} \geq 0$ are permutation matrices.*

- $Conv(\{\boldsymbol{z}_i^{[i]}\}) \supseteq E$
- $Conv(\{\boldsymbol{z}^{[i]}\}_i)^* \cap bd(E^*) = \{\lambda \boldsymbol{e}_k | \lambda \in \mathbb{R}, k = 1, ..., d_S\}$

As currently framed these conditions are sufficient, though we conjecture that they might also be necessary.

Partial Identifiability In practice, the identifiability of every single neuron tuning curve might be too much to ask. For example, you might hypothesise that your representation is made from two populations, each encoding a single variable. In section 8.C.3 we extend our results to this setting. We show that, in the studied setting, if the encoded variables are range independent (meaning that the range of one variable doesn't vary as you conditioned on the value of the other variable) then, even if the subpopulations mix, all optimal representations will contain a separate population encoding each variable.

8.5 A Nonlinear Normative Theory of ON-OFF Channel Coding

We now turn to a neural coding puzzle. It is observed in the retina that a single signal, the activity of bipolar cells, is split into two channels, ON and OFF, which each encode roughly half of the stimulus range (Euler et al., 2014; Sterling and Laughlin, 2015). This has been understood as an efficiency - splitting the encoding reduces the range of stimulus each neuron encodes, hence reducing the maximum and mean firing rate over the population, at a cost of more neurons (Sterling and Laughlin, 2015). Theoretical work has confirmed this, studying models using either an ON and OFF cell, or two ON/OFF cells, and shown that the combination of ON and OFF leads to greater efficiency (Gjorgjieva, Sompolinsky, and Meister, 2014). Further, the same signals are seen empirically in nonlinear networks trained to capture retinal-like activity (Ocko et al., 2018). Yet, ON-OFF splitting is not a ubiquitous phenomenon. For example, it appears that the V1 gabor-filter code does not have the same dichotomy. Sterling and Laughlin (2015) conjecture that this arises from the sparsity of the code: the retinal code is dense, making pathway splitting effective; the V1 code is significantly sparser, removing these savings.

Since most normative models are linear (Atick and Redlich, 1990; Dordek et al., 2016; Ocko et al., 2018; Olshausen and Field, 1996), the inherent nonlinearity of an ON-OFF rectification scheme makes this somewhat difficult to reason about. Gjorgjieva, Sompolinsky, and Meister (2014) cleanly sidestep this problem by directly comparing the two coding schemes, but this technique is poorly placed for normatively deriving the conditions

which determine whether ON-OFF or direct coding is optimal. We therefore make use of the nonlinearity in some of the convex problems introduced in section 8.2. In particular, we study the optimal nonnegative, energy efficient, nonlinear but linearly-decodable representation of a single variable. In this setting we confirm that the sparsity controls whether ON-OFF or direct coding is optimal, matching conjecture and data.

Problem Statement We consider a nonnegative representation from which a single variable can be decoded:

$$\boldsymbol{r}^T \boldsymbol{z}^{[i]} + \boldsymbol{b}_r = x^{[i]} \qquad \boldsymbol{z}^{[i]} \geq 0 \tag{8.9}$$

Subject to these constraints we minimise the energy loss:

$$\mathcal{L} = \langle ||\boldsymbol{z}^{[i]}||^2 \rangle_i + \lambda ||\boldsymbol{r}||^2 \tag{8.10}$$

Results In section 8.D, we find that the optimal representation comprises either two neurons splitting the encoding of x, an ON-OFF coding, or a single neuron, either ON or OFF, fig. 8.3. Intuitively, this follows from energy saving: to be linearly decodable, x must be embedded in the representation, but the nonlinearity means different regions can be split up and distributed across the neurons. The only way to use this to save energy is to split it into two parts - ON and OFF.

However, if there is a high probability of data at one extreme of the range of x then a single neuron response might be preferred. For example, imagine x is nonnegative but sparse, it equals 0 with some probability S. Then splitting into ON-OFF channels is detrimental, better to have a single ON channel that activates sparsely. We can calculate that a sparse single unit code is preferable using the following simple condition:

$$S > \frac{\langle x^{[i]} \rangle_i^2}{\langle (x^{[i]})^2 \rangle_i} \tag{8.11}$$

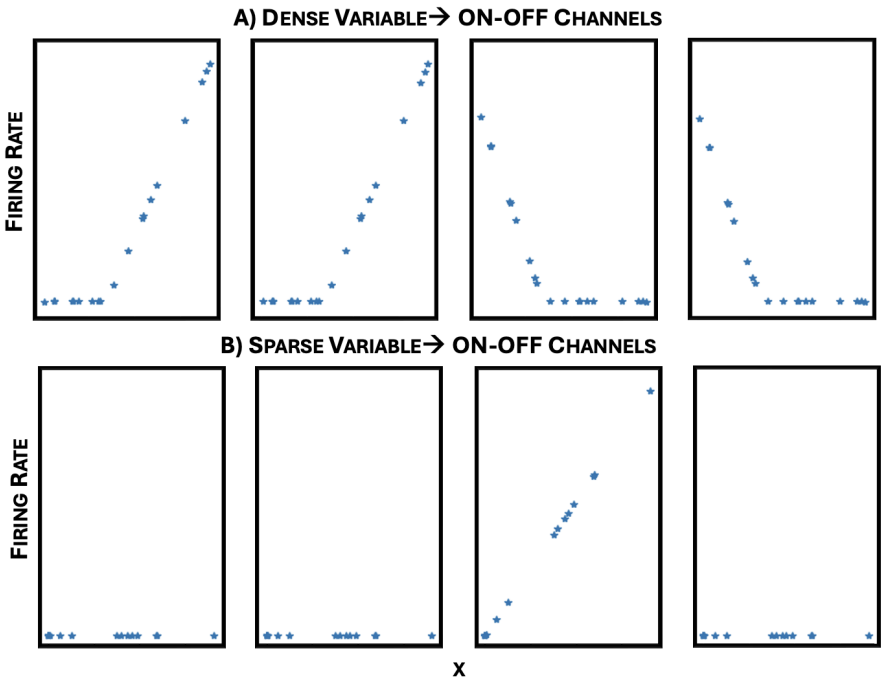


Figure 8.3: We numerically optimise a nonnegative, affine-decodable representation of a single variable. Its encoding is split into either **(A)** an OFF and ON channel, or **(B)** a single ON channel, depending on sparsity.

8.6 Discussion

In this work we showed that a variety of interesting representational optimisation problems could be written as convex optimisations over the representational similarity matrix. We then used this to derive a tight matrix factorisation identifiabiltiy result section 8.3, to understand the link between tuning curves and computation section 8.4, and to answer a coding puzzle regarding ON and OFF channels section 8.5. We think this flexible framework will be useful to others working in this area. We now review limitations, related, and future work.

Literature: Convex Analysis & Neuroscience While normative theorising about representations is standard, to the best of our knowledge, Sengupta et al. (2018) are the only work to use convexity in the way we have here. However, there are other settings where convexity has emerged as a model of a neural system (rather than via a tool). For example, beautiful paper shows that the behaviour of spiking neural neural networks can be understood as implementing a convex optimisation Mancoo, Keemink, and Machens (2020), and extensions have discussed modelling of nonconvex functions using low-rank networks (Podlaski and Machens, 2024).

Literature: Convex Analysis & Neural Networks A lot of work has, as here, reframed the optimisation of neural networks as a convex optimisation. Earlier approaches studied the addition of single neurons to optimised networks and found this was convex (Bach, 2017; Bengio et al., 2005). Neural tangent kernel approaches showed that, in the infinite width, training a neural network was convex, but this limit can break many interesting representational learning phenomenon (Jacot, Gabriel, and Hongler, 2018). More recent work has shown that the entire standard neural network problem can be framed as a convex optimisation for 1-hidden layer ReLU networks (Ergen and Pilanci, 2021; Pilanci and Ergen, 2020). These results have been extensively developed and used to understand machine learning phenomenology (Sahiner et al., 2020; Zeger and Pilanci, 2025). These results are impressive, and could likely be used for neuroscience. They differ with respect to ours in the type of convexity problem, and the exactness of the match to an artificial neural network. Their problems correspond exactly to neural networks, while ours focus on properties of the representation (though, at times, are equivalent); further, using our simplifying assumptions we are able to write the problems as convex over the set of representational similarity matrices, which we have only seen in Sengupta et al. (2018).

Technical Limitation: Fewer Datapoints than Neurons Most pressingly, our work requires that the number of datapoints is larger than the number of neurons. This is constraining; however we see two rays of hope. First, if you make this assumption, then perform the optimisation and find that there was an optima that used only a small number of neuron, then this will be the optimal solution to the neuron-limited problem too. We saw this at work in section 8.5: we found that the optimal solution used, at most, two neurons, so regardless of the number of datapoints we didn't have to use many neurons. A second more fundamental approach would be a convex relaxation of this constraint. The neuron-number constraint is a pain because it constrains the rank of the representational similarity matrix, $\boldsymbol{Q}$, and rank constraints are non-convex. The tightest convex relaxation of this is to penalise the nuclear norm of the matrix $\boldsymbol{Q}$, the sum of the singular values. By adjusting this regularisation hyperparameter, different convex problems with different ranks could be derived, each of which correspond to the optimal solution with a different number of neurons.

Technical Limitation: Using Convex Optimisation Tools Given the framing as convex, it is natural to wonder if we can use convex optimisation tools like the interior point method to numerically find optimal codes, rather than using gradient descent. A roadblock towards this goal is the difficulty of constraining $\boldsymbol{Q}$ to be within the set of completely positive matrices. This set of matrices is not well understood, making completely-positive programming difficult (Dür, 2010). Instead, we could directly optimise the representation, $\boldsymbol{Z}$, where the constraints are simply positivity and things like it. This will recover optimal $\boldsymbol{Z}$ up to orthogonal rotations, and the conditions in section 8.4 can be used to examine whether the neural tuning curves so discovered are likely to be robust.

Technical Limitation: Scattering Conditions A final technical concern is that verifying sufficient scattering conditions is often NP-hard. Fortunately there is work towards this goal (Gillis and Luce, 2024).

Neural Limitation: No Recurrence Our model is pleasingly flexible, allowing nonlinear reasoning that has thus far been tough in neural theory. However, our optimism should be tempered. The problems presented here are, in the grand scheme of things, relatively simple. We are not (yet) talking about the hierarchy of hidden layers of an interesting vision network, for example. Despite this, we remain hopeful that the tools presented here can be extended to many interesting cases. One obvious short-coming is our focus on representations just encoding variables, rather than computing with them. Our recent - the efficient computing hypothesis - has sought to extend these efficient coding approaches to populations of neurons performing interesting computations, beyond just nice formatings. Pleasingly these can be written as relatively simple, though non-convex, representational optimisation problems, and have proven effective at modelling both grid cells part I, and working memory representations in prefrontal cortex part III. We hope that convex relaxations or variable changes can permit the extension of these approaches to aid in the uncovering of real computational motifs in neural representations.

Neural Commentary: Linking Function and Representation Recent work has posited that representation and function are 'fundamentally disassociated', raising concerns about the basic methodology of neuroscience (Braun, E. Grant, and A. M. Saxe, 2025). However, we draw exactly opposing conclusion for

techniques using the RSA, which we then further defend with results on tuning-curve identifiability section 8.4. There will always be a huge variety of neural networks that implement the same function. For example, just add a bank of neurons to every hidden layer with zero output weights and you can arbitrarily change the hidden layer representations while leaving the function of the network unchanged. We do not find this worrying. Just because there is another neural network that performs a function doesn't mean the brain might implement it. Under a normative perspective the brain should minimise the use of energy, and such dead neurons are clearly a waste. With a meaningful set of energy constraints we and Braun, E. Grant, and A. M. Saxe (2025) both find that many interesting optimisation problems posses uniquely optimal representational similarity matrices. Further, we present results that show that even the single neuron tuning curves can often be reflective of the underlying computation. As such, we take a much more optimistic view: if we are able to correctly frame the computation and constraints, i.e. the optimisation problem, then it seems we often can tightly couple neural function and representation. This kind of normative procedure already seems to be working quite well, e.g. in understanding grid cells (Will Dorrell et al., 2023; Schaeffer, Khona, Ma, et al., 2023), and we have no reason to think the same approaches might not generalise to other neural circuits as we understand their function better.

Conclusions As such, we develop useful tools for reasoning over efficient coding arguments that we hope the field will find useful.

Technical Appendices: Convex Neural Optimisations

8.A A Family of Convex Objectives, Constraints, and Problems

In this section we demonstrate that a suite of interesting problems can be rewritten as convex optimisation problems over the set of representational similarity matrices. In this we are inspired by (Sengupta et al., 2018) which, to our knowledge, is the only other result of this sort. Their case considers nonnegative similarity matching. We broaden this to include (nonnegative) linear neural networks, and autoencoders using either a ReLU, unconstrained, or no nonlinearity. These results will later permit us to derive the first necessary and sufficient condition for regularised semi-nonnegative matrix factorisation; an identifiability condition for single neuron tuning curves in this family of problems.

We study a family of optimisation problems; each of which we reframe as optimisations over the set of representational similarity. These problems are made by composing a set of objectives and constraints. Since both the sum of convex functions, and the intersection of convex sets, are convex (Boyd and Vandenberghe, 2004) we show the convexity of each component objectives or constraints, which we can then combine to show convexity of a large set of problems. We begin by describing the family of problems, then we study convexity of each constraint set, before finally studying the convexity of the objectives.

8.A.1 Family of Problems

We consider optimising a representation, $\boldsymbol{Z} \in \mathbb{R}^{d_Z \times T}$, where d_z is the number of neurons, which we will when needed take to be very large, and T is the number of datapoints. We will prove that the set of representational similarity matrices that satisfy the following constraints are convex (some of them are mutually exclusive).

- C1 Nonnegativity: $\boldsymbol{Z} \geq 0$
- C2 Full rank affine relationship between representation and input dataset such that $\boldsymbol{Y}$ can be decoded, $\boldsymbol{X} \in \mathbb{R}^{d_x \times T}$: $\boldsymbol{W}_{\text{in}}\boldsymbol{X} + \boldsymbol{b}_{\text{in}}\boldsymbol{1}^T = \boldsymbol{Z}$, $\text{rank}(\boldsymbol{W}_{\text{in}}) = d_x$.
- C3 Perfect affine decodability of a dataset, $\boldsymbol{Y} \in \mathbb{R}^{d_y \times T}$: $\boldsymbol{W}_{\text{out}}\boldsymbol{Z} + \boldsymbol{b}_{\text{out}}\boldsymbol{1}^T = \boldsymbol{Y}$.
- C4 A ReLU relationship between representation and input dataset, $\boldsymbol{X} \in \mathbb{R}^{d_x \times T}$: $\boldsymbol{Z} = \text{ReLU}(\boldsymbol{W}_{\text{in}}\boldsymbol{X} + \boldsymbol{b}_{\text{in}}\boldsymbol{1}^T)$
- C5 A firing rate constraint: $[\boldsymbol{Z}^T\boldsymbol{Z}]_{ii} \leq k \forall i$.

Further, we will show that the following objectives are convex functions of representational similarity matrices.

- O1 An L2 activity loss: $||\boldsymbol{Z}||_F^2$
- O2 An L1 activity loss: $\sum_d ||\boldsymbol{z}_d||_1^2$, when the representation is nonnegative (C1)
- O3 An L2 input weight loss when the relationship is affine (C3 not C4): $||\boldsymbol{W}_{\text{in}}||_F^2$
- O4 An L2 output weight loss when the data is perfectly affine decodable (C3): $||\boldsymbol{W}_{\text{out}}||_F^2$
- O5 A similarity matching objective for a dataset $\boldsymbol{X}$, as in (Sengupta et al., 2018): $-\text{Tr}[(\boldsymbol{X}^T\boldsymbol{X} - \alpha\boldsymbol{1}\boldsymbol{1}^T)\boldsymbol{Z}^T\boldsymbol{Z}]$.
- O6 Nonlinear similarity matching with input similarity $\chi \in \mathbb{R}^{T \times T}$ and elementwise exponentiation: $-\text{Tr}[\chi e^{\boldsymbol{Z}^T\boldsymbol{Z}}]$.

Some examples of problems you can create from this menu:

1. C1, C2, C3, & O1, O3, O4: The positive affine autoencoder, as considered in section 5.1. Then input data is output data: $\boldsymbol{Y} = \boldsymbol{X}$.
2. C2, C3, & O3, O4: A regularised linear neural network, as in (A. M. Saxe, McClelland, and Ganguli, 2019).
3. C1, C2, C3, & O3, O4: A regularised linear neural network with a positivity constraint..
4. C1, C5, & O5: Nonnegative similarity matching, as in Sengupta et al. (2018).
5. C1, C3, C4, & O2, O4: A regularised one-layer ReLU neural network; such as a sparse autoencoder. Here too $\boldsymbol{Y} = \boldsymbol{X}$.
6. C1, C3, & O2, O4: A regularised linearly decodable representation, such as an unconstrained sparse autoencoder.

8.A.2 Convex Feasible Sets

We now show the convexity of the feasible set of representational similarity matrices for each constraint.

C1: Positive Representations Following (Sengupta et al., 2018)), the set of dot product matrices with unconstrained ranks formed from vectors that are themselves positive ($\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$ for $\boldsymbol{Z} \geq 0$) form a convex set called the set of completely positive matrices (Berman and Shaked-Monderer, 2003). As long as we allow many neurons so there is no rank constraint, we're good.

C2: Affine Input First let's rewrite the affine constraint more simply. Let's define:

$$\boldsymbol{X}' = \begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{1}\end{bmatrix} \qquad \boldsymbol{W}'_{\text{in}} = \begin{bmatrix}\boldsymbol{W}_{\text{in}} & \boldsymbol{b}_{\text{in}}\end{bmatrix} \quad \text{such that} \quad \boldsymbol{Z} = \boldsymbol{W}'_{\text{in}}\boldsymbol{X}' \tag{8.12}$$

Now, take two dot product matrices formed from representations that are affine functions of the inputs: $\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$, $\boldsymbol{Z} = \boldsymbol{W}'_{\text{in}}\boldsymbol{X}'$, and $\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{Z}}^T\tilde{\boldsymbol{Z}}$, $\tilde{\boldsymbol{Z}} = \tilde{\boldsymbol{W}}'_{\text{in}}\boldsymbol{X}'$. Take their convex combination:

$$\lambda\boldsymbol{Q} + (1-\lambda)\tilde{\boldsymbol{Q}} = \boldsymbol{X}'^T(\lambda\boldsymbol{W}'^T_{\text{in}}\boldsymbol{W}'_{\text{in}} + (1+\lambda)\tilde{\boldsymbol{W}}'^T_{\text{in}}\tilde{\boldsymbol{W}}'_{\text{in}})\boldsymbol{X}' \quad \lambda \in [0,1] \tag{8.13}$$

Both $\boldsymbol{W}'^T_{\text{in}}\boldsymbol{W}'_{\text{in}}$ and $\tilde{\boldsymbol{W}}'^T_{\text{in}}\tilde{\boldsymbol{W}}'_{\text{in}}$ are positive semi-definite matrices, therefore their convex combination will be as well. Further any positive semidefinite matrix can be decomposed into two parts, therefore we can write $\lambda\boldsymbol{W}'^T_{\text{in}}\boldsymbol{W}'_{\text{in}} + (1+\lambda)\tilde{\boldsymbol{W}}'^T_{\text{in}}\tilde{\boldsymbol{W}}'_{\text{in}} = \hat{\boldsymbol{W}}'^T_{\text{in}}\hat{\boldsymbol{W}}'_{\text{in}}$ for some matrix $\hat{\boldsymbol{W}}'_{\text{in}}$. Hence

$$\lambda\boldsymbol{Q} + (1-\lambda)\tilde{\boldsymbol{Q}} = \boldsymbol{X}'^T\hat{\boldsymbol{W}}'^T_{\text{in}}\hat{\boldsymbol{W}}'_{\text{in}}\boldsymbol{X}' = (\hat{\boldsymbol{W}}'_{\text{in}}\boldsymbol{X}')^T(\hat{\boldsymbol{W}}'_{\text{in}}\boldsymbol{X}') = \hat{\boldsymbol{Z}}^T\hat{\boldsymbol{Z}} \tag{8.14}$$

Hence the convex combination of two similarity matrices of affine functions of the input data will also be such a similarity matrix.

C3: Perfect Affine Decodability Let's begin by the easier problem of linear decodability. Denote with ρ the rank of the data $\boldsymbol{Y}$. For $\boldsymbol{Y}$ to be linearly decodable from $\boldsymbol{Z}$, $\boldsymbol{Z}$ must contain a ρ subspace that encodes the data, let's call it $\boldsymbol{A}$. The remaining $T-\rho$ dimensions have to encode activity that is orthogonal to the demeaned patterns, $\boldsymbol{Z}_\perp$: $\boldsymbol{Z}_\perp\boldsymbol{Y}^T = \boldsymbol{0}$.

$$\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{Y} + \boldsymbol{Z}_\perp \qquad \boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z} \tag{8.15}$$

We now make a convex combination of two representational similarity matrices:

$$\boldsymbol{Q}_\lambda = \lambda\boldsymbol{Q} + (1-\lambda)\tilde{\boldsymbol{Q}} \tag{8.16}$$

We therefore need to test whether the resulting representational similarity matrix has been made from a representation from which $\boldsymbol{Y}$ is linearly decodable. For any representation $\boldsymbol{Z}'$, since $d_Z > T$, if $\boldsymbol{Z}'\boldsymbol{Y}^T$ is rank ρ then $\boldsymbol{Y}$ can be linearly decoded from $\boldsymbol{Z}'$. Therefore, similarly, if $\boldsymbol{Y}\boldsymbol{Q}\boldsymbol{Y}^T$ is rank ρ, $\boldsymbol{Q}$ is a representational similarity matrix made from a linearly decodable representation.

We can now test the convex combination and to see if it is affine decodable:

$$\boldsymbol{Y}\boldsymbol{Q}_\lambda\boldsymbol{Y}^T = \underbrace{\lambda\boldsymbol{Y}\boldsymbol{Q}\boldsymbol{Y}^T}_{\text{rank}=\rho} + \underbrace{(1-\lambda)\boldsymbol{Y}\tilde{\boldsymbol{Q}}\boldsymbol{Y}^T}_{\text{rank}=\rho} \tag{8.17}$$

The rank of $\boldsymbol{Y}\boldsymbol{Q}_\lambda\boldsymbol{Y}^T$ is at most ρ, but since both $\lambda\boldsymbol{Y}\boldsymbol{Q}\boldsymbol{Y}^T$ and $(1-\lambda)\boldsymbol{Y}\tilde{\boldsymbol{Q}}\boldsymbol{Y}^T$ are positive definite rank ρ matrices, the rank of their sum is greater than or equal to ρ. Therefore the rank of $\boldsymbol{Y}\boldsymbol{Q}_\lambda\boldsymbol{Y}^T$ is ρ, and the representational similarity matrix is made from a representation that is linearly decodable. The set of representational similarities of linearly decodable representations form a convex set.

Finally, we need to make the step back to affine decodable. For $\boldsymbol{Z}$ to be linearly decodable the representation with a constant must be linearly decodable:

$$\boldsymbol{Z}_c = \begin{bmatrix}\boldsymbol{Z}\\ \boldsymbol{1}\end{bmatrix} \in \mathbb{R}^{(N+1)\times T} \tag{8.18}$$

The representational similarities of these two representations are closely related:

$$\boldsymbol{Q}_c = \boldsymbol{Z}_c^T\boldsymbol{Z}_c = \boldsymbol{Q} + \boldsymbol{1}\boldsymbol{1}^T \tag{8.19}$$

So, as before, in order to be linearly decodable, $\boldsymbol{Y}\boldsymbol{Q}_c\boldsymbol{Y}^T$ has to be rank ρ. One rank can come from the constant, $\boldsymbol{1}\boldsymbol{1}^T$ but that can only fit the mean of the data. If we create the demeaned data, $\bar{\boldsymbol{Y}}$ our condition is either unchanged if the data had mean zero to begin one, or becomes that $\bar{\boldsymbol{Y}}\boldsymbol{Q}\bar{\boldsymbol{Y}}$ is rank $\rho-1$ if the data had a mean. If two $\boldsymbol{Q}$ matrices have either of these properties, their convex combination will as well, so the set of affine decodable representational similarity matrices is a convex set.

C4: ReLU Input Let's say you have two representation that are ReLU and affine related to their inputs:

$$\boldsymbol{Z} = \text{ReLU}(\boldsymbol{W}_{\text{in}}\boldsymbol{X} + \boldsymbol{b}_{\text{in}}\boldsymbol{1}^T) \qquad \tilde{\boldsymbol{Z}} = \text{ReLU}(\tilde{\boldsymbol{W}}_{\text{in}}\boldsymbol{X} + \tilde{\boldsymbol{b}}_{\text{in}}\boldsymbol{1}^T) \tag{8.20}$$

With their respective representational similarity matrices, $\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$ and $\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{Z}}^T\tilde{\boldsymbol{Z}}$. Given enough neurons we can always create another representation $\boldsymbol{Z}_\lambda$ with representational similarity matrix $\lambda\boldsymbol{Q} + (1-\lambda)\tilde{\boldsymbol{Q}}$:

$$\boldsymbol{Z}_\lambda = \text{ReLU}\left(\begin{bmatrix} \sqrt{\lambda}\boldsymbol{W}_{\text{in}} \\ \sqrt{1-\lambda}\tilde{\boldsymbol{W}}_{\text{in}} \end{bmatrix} \boldsymbol{X} + \begin{bmatrix} \sqrt{\lambda}\boldsymbol{b}_{\text{in}}\boldsymbol{1}^T \\ \sqrt{1-\lambda}\tilde{\boldsymbol{b}}_{\text{in}}\boldsymbol{1}^T \end{bmatrix} \right) = \begin{bmatrix} \sqrt{\lambda}\boldsymbol{Z} \\ \sqrt{1-\lambda}\tilde{\boldsymbol{Z}} \end{bmatrix} \tag{8.21}$$

C5: Constrained firing rate norm This is simple, if the diagonals of both $\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$ and $\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{Z}}^T\tilde{\boldsymbol{Z}}$ are below some constant k, then the same will be true for convex combinations of $\boldsymbol{Q}$ and $\tilde{\boldsymbol{Q}}$

8.A.3 Convex Objectives

Here we show the convexity of each proposed objective.

O1: L2 Activity The activity loss can be easily rewritten:

$$\frac{1}{T}||\boldsymbol{Z}||_F^2 = \frac{1}{T}\,\text{Tr}[\boldsymbol{Z}^T\boldsymbol{Z}] = \frac{1}{T}\,\text{Tr}[\boldsymbol{Q}] \tag{8.22}$$

which since it is a linear function of $\boldsymbol{Q}$ is clearly a convex function.

O2: L1 Activity We define a particular form of the L1 activity loss:

$$\sum_{d=1}^{d_z} ||\boldsymbol{z}_d||_1^2 = \sum_d (\sum_t |z_d^{[t]}|)^2 = \sum_{d,t,t'} |z_d^{[t]}||z_d^{[t']}| \tag{8.23}$$

However, if the representation is nonnegative we can drop the absolute value. Then we can see this is a linear and therefore convex function of $\boldsymbol{Q}$:

$$\sum_{d,t,t'} z_d^{[t]} z_d^{[t']} = \sum_{t,t'} \boldsymbol{z}^{[t],T}\boldsymbol{z}^{[t']} = \sum_{t,t'} Q_{t,t'} = \boldsymbol{1}^T\boldsymbol{Q}\boldsymbol{1} \tag{8.24}$$

O3: L2 Input Weights We know that $\boldsymbol{Z} = \boldsymbol{W}_{\text{in}}\boldsymbol{X} + \boldsymbol{b}_{\text{in}}\boldsymbol{1}^T$. The work of mapping the mean between these two representations can be at least partly done by the bias, correcting any errors left by the linear map. Therefore the min-norm choice of linear map just maps the demeaned data to the demeaned representation:

$$\boldsymbol{W}_{\text{in}} = \bar{\boldsymbol{Z}}\bar{\boldsymbol{X}}^\dagger \tag{8.25}$$

This makes its frobenius norm:

$$\text{Tr}[\bar{\boldsymbol{X}}^\dagger\bar{\boldsymbol{X}}^{T,\dagger}\bar{\boldsymbol{Z}}^T\bar{\boldsymbol{Z}}] = \text{Tr}[\bar{\boldsymbol{X}}^\dagger\bar{\boldsymbol{X}}^{T,\dagger}(\boldsymbol{Z} - \boldsymbol{Z}\boldsymbol{1}\boldsymbol{1}^T)^T(\boldsymbol{Z} - \boldsymbol{Z}\boldsymbol{1}\boldsymbol{1}^T)] = \text{Tr}[\bar{\boldsymbol{X}}^\dagger\bar{\boldsymbol{X}}^{T,\dagger}\boldsymbol{Q}] \tag{8.26}$$

since $\bar{\boldsymbol{X}}\boldsymbol{1} = \boldsymbol{0}$, and the columns of $\bar{\boldsymbol{X}}^\dagger$ has the same span as the rows of $\bar{\boldsymbol{X}}$, so $\boldsymbol{1}^T\bar{\boldsymbol{X}}^\dagger = \boldsymbol{0}$. This is a linear, and therefore convex, function of $\boldsymbol{Q}$.

O4: Best Fitting Readout Weights Similarly, the min-norm readout linear map has to map the demeaned representation to the demeaned data:

$$\boldsymbol{W}_{\text{out}}\bar{\boldsymbol{Z}} = \bar{\boldsymbol{Y}} \tag{8.27}$$

The min-norm choice is given by the pseudoinverse:

$$\boldsymbol{W}_{\text{out}} = \bar{\boldsymbol{Y}}\bar{\boldsymbol{Z}}^\dagger \tag{8.28}$$

Since the demeaned data is linearly decodable from the representation by assumption, the representation must contain a ρ-dimensional subspace encoding the demeaned data, where ρ is the rank of $\bar{\boldsymbol{Y}}$. To make life easier let's map $\bar{\boldsymbol{Y}}$ into a ρ dimensional space so that it has full row rank: $\bar{\boldsymbol{Y}}_\rho = \boldsymbol{B}\bar{\boldsymbol{Y}} \in \mathbb{R}^{\rho\times T}$. Then the representation must encode this:

$$\boldsymbol{Z} = \boldsymbol{A}\bar{\boldsymbol{Y}}_\rho + \boldsymbol{Z}_\perp \tag{8.29}$$

where $\boldsymbol{Z}_\perp$ is the rest of the activity that is perpendicular to $\bar{\boldsymbol{Y}}_\rho$.

The frobenius norm of the readout matrix is then:

$$||\boldsymbol{W}_{\text{out}}||_F^2 = \text{Tr}[\bar{\boldsymbol{Y}}_\rho(\boldsymbol{A}\bar{\boldsymbol{Y}} + \boldsymbol{Z}_\perp)^\dagger(\boldsymbol{A}\bar{\boldsymbol{Y}} + \boldsymbol{Z}_\perp)^{\dagger,T}\bar{\boldsymbol{Y}}_\rho^T] = \text{Tr}[\bar{\boldsymbol{Y}}_\rho\bar{\boldsymbol{Y}}_\rho^\dagger\boldsymbol{A}^\dagger\boldsymbol{A}^{\dagger,T}\bar{\boldsymbol{Y}}_\rho^{\dagger,T}\bar{\boldsymbol{Y}}_\rho^T] \tag{8.30}$$

where we used the orthogonality of $\boldsymbol{Z}_\perp\boldsymbol{Y}^T = \boldsymbol{0}$, and the fact that the columns of a matrix and the rows of its pseudoinverse share the same span. Further, since $\bar{\boldsymbol{Y}}_\rho$ has full row rank $\bar{\boldsymbol{Y}}\boldsymbol{Y}^\dagger$ is the identity matrix, so life is easy:

$$||\boldsymbol{W}_{\text{out}}||_F^2 = \text{Tr}[\boldsymbol{A}^\dagger\boldsymbol{A}^{\dagger,T}] = \text{Tr}[(\boldsymbol{A}\boldsymbol{A}^T)^{-1}] = \text{Tr}[\boldsymbol{O}_A^{-1}] \tag{8.31}$$

The trace of the inverse of a positive semidefinite matrix like $\boldsymbol{O}_A$ is a convex function of $\boldsymbol{O}_A$, so now we just have to work our way back to the representational similarity matrices $\boldsymbol{Q}$. This is simple because for every affine decodable $\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$ there is an associated $\boldsymbol{O}_A$ matrix, so a convex function on $\boldsymbol{O}_A$ matrices is a convex function on $\boldsymbol{Q}$ matrices.

Finally, we make one further claim. Since $\boldsymbol{Z}$ is affine decodable $\boldsymbol{O}_A$ has to be positive definite, rather than just semidefinite. Over the set of positive definite matrices $\text{Tr}[\boldsymbol{O}_A^{-1}]$ is a strictly[1] convex function of $\boldsymbol{O}_A$. The strict convexity means that globally optimal $\boldsymbol{Q}$ matrices must have the same $\boldsymbol{O}_A$.

O5: Similarity Matching Objective In terms of the representational similarity this is:

$$\text{Tr}[(\boldsymbol{X}^T\boldsymbol{X} - \alpha\boldsymbol{1}\boldsymbol{1}^T)\boldsymbol{Q}] \tag{8.32}$$

This is a linear function of $\boldsymbol{Q}$ so is convex.

O6: Nonlinear Similarity Matching We consider the nonlinear similarity matching objective:

$$\text{Tr}[\chi e^{\boldsymbol{Z}^T\boldsymbol{Z}}] = \text{Tr}[\chi e^{\boldsymbol{Q}}] \tag{8.33}$$

for a positive-definite input similarity kernel $\chi \in \mathbb{R}^{T\times T}$. We can take the second derivative of this with respect to $\boldsymbol{Q}$:

$$\frac{\partial^2\mathcal{L}}{\partial Q_{ab}\partial Q_{cd}} = \delta_{ac}\delta_{bd}e^{Q_{ab}}\chi_{ab} \tag{8.34}$$

Schoenberg (1942) showed that elementwise functions of a positive definite matrix are positive definite if the function is analytic with all positive coefficients. This tells us that the elementwise exponential of a postive semidefinite matrix is positive semidefinite. Similarly, the elementwise product of two postive semi-definite matrices is postive semi-definite. Hence this derivative is positive and the function is convex.

[1] $f(x)$ is a convex function is for all $\lambda \in [0,1]$ and pairs of inputs x and x' $f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda)f(x')$. If the equality is only achieved when $x = x'$ or $\lambda = 0$ or 1, it is called strictly convex.

8.B Necessary and Sufficient Identifiability under Bio-Constraints

Let's recall our optimisation problem:

Problem 2 (Biological Linear Autoencoding of Linearly Mixed Sources). *Let $\boldsymbol{s} \in \mathbb{R}^{d_s}$, $\boldsymbol{x} \in \mathbb{R}^{d_x}$, $\boldsymbol{z} \in \mathbb{R}^{d_z}$, $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{d_z \times d_x}$, $\boldsymbol{b}_{\text{in}} \in \mathbb{R}^{d_x}$, $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{d_x \times d_z}$, $\boldsymbol{b}_{\text{out}} \in \mathbb{R}^{d_x}$, and $\boldsymbol{A} \in \mathbb{R}^{d_x \times d_s}$ where $d_z > d_s$ and $d_x \geq d_s$, and $\boldsymbol{A}$ is rank d_s. We seek the solution to the following the constrained optimization problem.*

$$\begin{aligned} \min_{\boldsymbol{W}_{\text{in}}, \boldsymbol{b}_{\text{in}}, \boldsymbol{W}_{\text{out}}, \boldsymbol{b}_{\text{out}}} \quad & \left\langle ||\boldsymbol{z}^{[i]}||_2^2 \right\rangle_i + \lambda \left(||\boldsymbol{W}_{\text{in}}||_F^2 + ||\boldsymbol{W}_{\text{out}}||_F^2 \right) \\ \text{s.t.} \quad & \boldsymbol{z}^{[i]} = \boldsymbol{W}_{\text{in}} \boldsymbol{x}^{[i]} + \boldsymbol{b}_{\text{in}},\ \boldsymbol{x}^{[i]} = \boldsymbol{W}_{\text{out}} \boldsymbol{z}^{[i]} + \boldsymbol{b}_{\text{out}},\ \boldsymbol{z}^{[i]} \geq 0, \end{aligned} \tag{8.35}$$

where i indexes a finite set of samples of $\boldsymbol{s}$, and $\boldsymbol{x}^{[i]} = \boldsymbol{A}\boldsymbol{s}^{[i]}$.

When will this optimisation problem produce a modular solution? In other words, when will each latent neuron be a function of only one source: $\boldsymbol{z}_n^{[i]} = d_n(s_k^{[i]} - \min_i s_k^{[i]})$? We will present tight conditions on the dataset $\{\boldsymbol{s}^{[i]}\}_i$ and mixing matrix $\boldsymbol{A}$ such that if this algorithm is properly optimised, the neurons will recover the sources. This is known as an identifiability result: a statement about what has to be true about the data such that an algorithm will, in principle, succeed in extracting the groundtruth sources.

Tight Scattering Condition In this work we show that a necessary and sufficient condition for the optimal biological linear autoencoder to recover the true sources is the following tight scattering condition:

Definition 8.B.1 (Tight Scattering). *First, generate the de-meaned sources: $\bar{\boldsymbol{S}} = \boldsymbol{S} - \boldsymbol{S}\boldsymbol{1}\boldsymbol{1}^T \in \mathbb{R}^{d_S \times T}$ with elements $\bar{S}_{dt} = \bar{s}_d^{[t]}$. Assume without loss of generality that $|\min_t \bar{s}_d^{[t]}| \leq \max_t \bar{s}_d^{[t]}$ for each dimension d (if this is not satisfied simply redefine this source as $-\boldsymbol{s}_d$), then construct the diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{d_S \times d_S}$ and the symmetric matrix $\boldsymbol{F}^{d_S \times d_S}$:*

$$D_{jj} = \sqrt[4]{\frac{\lambda(\boldsymbol{A}^T\boldsymbol{A})_{jj}}{\langle(\bar{s}_j^{[i]})^2\rangle_i + (\min_i \bar{s}_j^{[i]})^2 + \lambda((\boldsymbol{A}^T\boldsymbol{A})^{-1})_{jj}}} \qquad \boldsymbol{F} = \lambda \boldsymbol{D}^{-2}(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{D}^{-2} - \lambda(\boldsymbol{A}^T\boldsymbol{A})^{-1} - \bar{\boldsymbol{S}}\bar{\boldsymbol{S}}^T \tag{8.36}$$

Then use these matrices to construct the following set:

$$E = \{\boldsymbol{x} | \boldsymbol{x}^T \boldsymbol{F}^{-1} \boldsymbol{x} = 1\} \tag{8.37}$$

Then $\boldsymbol{S}$ is tightly scattered with respect to a generating matrix $\boldsymbol{A}$ if the following conditions hold:

- *Conv$(\bar{\boldsymbol{S}}) \supseteq E$*
- *Conv$(\bar{\boldsymbol{S}})^* \cap bdE^* = \{\lambda \boldsymbol{e}_k | \lambda \geq 0, k = 1, ..., d_S\}$ where the $*$ denotes the dual cone, and bd is the boundary.*

These two conditions clearly relate to sufficient scattering, but are adaptive to the structure of $\boldsymbol{A}$. Further, it is easy to check that the diagonal elements of $\boldsymbol{F}$ are $F_{jj} = (\min_i[\bar{s}_j^{[i]}])^2$, which will be useful later.

Main Result Our main theorem is that tight scattering is a necessary and sufficient condition for biological linear autoencoders to recover linearly mixed sources.

Theorem 7 (Identifiability of Biological Linear Autoencoders). *Given a dataset $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$, if the matrix $\bar{\boldsymbol{S}}$ is tightly scattered with respect to $\boldsymbol{A}$ then the optimal biological linear autoencoder recovers the sources - each neuron's activity is an affine function of one source and every source has at least one neuron encoding it:*

$$\boldsymbol{z}_n^{[i]} = d_n(s_n^{[i]} - \min_i[s_n^i]) \tag{8.38}$$

Our proof takes the following logic: we find the optimal modular solution (section 8.B.1), then we locally perturb into the nearby mixed representations. We derive the above conditions as those that determine when the optimal modular solution is in fact also a local minima of the full representation, section 8.B.2. Since the problem is convex, section 8.2, it is also then the global minima. We then show that if these conditions hold there are no non-modular global minima, i.e. the globally minimum representations are all modular. Finally, we show that in the special case where the sources are mixed orthogonally we recover the results found by William Dorrell, K. Hsu, et al. (2025) (section 8.B.4).

8.B.1 Optimal Modular Solution

The best modular representation can be implemented by d_S neurons, one for each source. Let's first demean the sources, which doesn't change the problem (since we can add or remove a bias arbitrarily), but makes notation easier. Then:

$$\boldsymbol{z}_M^{[i]} = \sum_{j=1}^{d_S} d_j(\bar{s}_j^{[i]} - \min_i \bar{s}_j^{[i]})\boldsymbol{e}_j \tag{8.39}$$

Where $\boldsymbol{e}_j$ are the canonical basis vectors, d_j represents the strength of the encoding of each source, and the subscript M denotes the fact it is modular. Stacking the d_j elements into a diagonal matrix $\boldsymbol{D}$ we can derive each of the three loss terms. We know that:

$$\boldsymbol{z}_M^{[i]} = \boldsymbol{W}_{\text{in}}\boldsymbol{x}^{[i]} + \boldsymbol{b}_{\text{in}} = \boldsymbol{W}_{\text{in}}\boldsymbol{A}\bar{\boldsymbol{s}}^{[i]} + \boldsymbol{b}_{\text{in}} = \boldsymbol{D}\bar{\boldsymbol{s}}^{[i]} + \boldsymbol{b}_{\text{in}} \tag{8.40}$$

Hence the min L2 norm weight matrices are:

$$\boldsymbol{W}_{\text{in}} = \boldsymbol{D}\boldsymbol{A}^{\dagger} \qquad \boldsymbol{W}_{\text{out}} = \boldsymbol{W}_{\text{in}}^{\dagger} = \boldsymbol{A}\boldsymbol{D}^{-1} \tag{8.41}$$

where † denotes the pseudoinverse. Hence:

$$||\boldsymbol{W}_{\text{in}}||_F^2 = \text{Tr}[\boldsymbol{W}_{\text{in}}^T\boldsymbol{W}_{\text{in}}] = \text{Tr}[\boldsymbol{D}^2\boldsymbol{A}^{\dagger}(\boldsymbol{A}^{\dagger})^T] = \text{Tr}[\boldsymbol{D}^2\boldsymbol{O}_A^{-1}] \tag{8.42}$$

$$||\boldsymbol{W}_{\text{out}}||_F^2 = \text{Tr}[\boldsymbol{D}^{-2}\boldsymbol{O}_A] \tag{8.43}$$

Where we've defined $\boldsymbol{O}_A = \boldsymbol{A}^T\boldsymbol{A}$, and then $\boldsymbol{A}^{\dagger}(\boldsymbol{A}^{\dagger})^T = \boldsymbol{O}_A^{-1}$, as can be seen using the SVD.

Denote with α_j^2 the diagonal elements of $\boldsymbol{O}_A$, positive numbers representing how strongly a particular source is encoded in the data. Further, denote with $(\alpha'_j)^2$ the diagonal elements of $\boldsymbol{O}_A^{-1}$, these represent the lengths of the 'pseudoinverse vectors'. If the encodings of each source are orthogonal they are just the inverse of the encoding sizes, α_j, if not, they also tell you how 'entangled' a source's representation is. In terms of these variables we can write the whole loss as:

$$\mathcal{L} = \sum_{j=1}^{d_s} d_j^2[\langle(s_j^{[i]})^2\rangle_i + (\min_i s_j^{[i]})^2 + \lambda(\alpha'_j)^2] + \frac{\lambda\alpha_j^2}{d_j^2} \tag{8.44}$$

Then the optimal choice of encoding sizes is:

$$(d_j^*)^4 = \frac{\lambda\alpha_j^2}{\langle(s_j^{[i]})^2\rangle_i + (\min_i s_j^{[i]})^2 + \lambda(\alpha'_j)^2} \tag{8.45}$$

This corresponds to the elements of the matrix $\boldsymbol{D}$ from the definition of tight scattering, definition 8.3.1, hence why we gave them the same name.

8.B.2 Tight Scattering is a Necessary and Sufficient Condition for the Modular Representation to be a Global Minima

Now let's just go balls to the wall and take the derivative of the whole loss with respect to the linear map that relates the neurons to the sources, while being careful to preserve the positivity of the representation.

$$\boldsymbol{z}^{[i]} = \boldsymbol{W}\bar{\boldsymbol{s}}^{[i]} - \min_i[\boldsymbol{W}\bar{\boldsymbol{s}}^{[i]}] \tag{8.46}$$

where the minima is taken neuron-wise. Then the min-norm weight matrices are: $\boldsymbol{W}_{\text{in}} = \boldsymbol{W}\boldsymbol{A}^{\dagger}$, $\boldsymbol{W}_{\text{out}} = \boldsymbol{A}\boldsymbol{W}$, so, defining the covariance matrix $\boldsymbol{\Sigma} = \langle\bar{\boldsymbol{s}}^{[i]}\bar{\boldsymbol{s}}^{[i],T}\rangle_i$, we can write the loss as:

$$\mathcal{L}(\boldsymbol{W}) = \lambda\,\text{Tr}[\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{O}_A^{-1}] + \lambda\,\text{Tr}[(\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{O}_A] + \text{Tr}[\boldsymbol{\Sigma}\boldsymbol{W}^T\boldsymbol{W}] + \sum_n(\min_i[\boldsymbol{w}_n\bar{\boldsymbol{s}}^{[i]}])^2 \tag{8.47}$$

where the min is now just the min of a set of scalars, and we've denoted with $\boldsymbol{w}_n$ the nth row of $\boldsymbol{W}$. Then the (sub)derivative is:

$$\frac{1}{2}\frac{\partial\mathcal{L}}{\partial\boldsymbol{W}} = \lambda\boldsymbol{W}\boldsymbol{O}_A^{-1} - \lambda\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{O}_A(\boldsymbol{W}^T\boldsymbol{W})^{-1} + \boldsymbol{W}\boldsymbol{\Sigma} + \sum_n \min_i[\boldsymbol{w}_n^T\boldsymbol{s}^{[i]}]\frac{\partial\min_i[\boldsymbol{w}_n^T\boldsymbol{s}^{[i]}]}{\partial\boldsymbol{W}} \tag{8.48}$$

Let's evaluate this at the optimal modular representation, which with a slight abuse of notation is $\boldsymbol{W} = \boldsymbol{D}$ (previously $\boldsymbol{D} \in \mathbb{R}^{d_S \times d_S}$, but now we'll pad it with zeros so it is $\boldsymbol{D} \in \mathbb{R}^{N \times d_S}$ like $\boldsymbol{W}$). The biggest immediate change is in the minima term; since:

$$\text{for} \quad n < d_S \quad \boldsymbol{w}_n = \boldsymbol{e}_n d_n \qquad \text{so} \quad \min_i[\boldsymbol{w}_n^T \bar{\boldsymbol{s}}^{[i]}] = d_n \min_i[\bar{s}_n^{[i]}] \tag{8.49}$$

Further, the subderivative of the minima of a set of convex functions is within the convex hull of the derivatives of the functions that achieve that minima:

$$f(x) = \min_i f_i(x) \qquad \partial f(x) \in \text{Convex Hull}(\{\partial f_i(x) | f_i(x) = f(x)\}) \tag{8.50}$$

So we know that:

$$\frac{\partial \min_i[\boldsymbol{w}_n^T \boldsymbol{s}^{[i]}]}{\partial \boldsymbol{w}_{n'}} = \delta_{n,n'} \boldsymbol{a}_n \qquad \boldsymbol{a}_n \in \text{Convex Hull}(\{\boldsymbol{s}^{[i]} | s_n^{[i]} = \min_j s_n^{[j]}\}) \tag{8.51}$$

i.e. the subdifferential vector $\boldsymbol{a}_n$ lives in the convex hull of the bounding datapoints in direction n. Now we want to know if the derivative is zero. Let's look at the gradient with respect to one row of the matrix $\boldsymbol{W}$, $\boldsymbol{w}_n$:

$$\frac{1}{2} \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_n}\bigg|_{\boldsymbol{W}=\boldsymbol{D}} = \lambda d_n \boldsymbol{e}_n^T \boldsymbol{O}_A^{-1} - \lambda d_n \boldsymbol{e}_n^T (\boldsymbol{D}^T \boldsymbol{D})^{-1} \boldsymbol{O}_A (\boldsymbol{D}^T \boldsymbol{D})^{-1} + d_n \boldsymbol{e}_n^T \boldsymbol{\Sigma} + d_n \min_i[s_n^{[i]}] \boldsymbol{a}_n \tag{8.52}$$

Therefore, an equivalent question to establishing whether this gradient is zero is the following question, is this vector in the convex hull of the axis-bounding points?

$$\frac{1}{-\min_i[s_n^{[i]}]} \underbrace{\left(\lambda (\boldsymbol{D}^T \boldsymbol{D})^{-1} \boldsymbol{O}_A (\boldsymbol{D}^T \boldsymbol{D})^{-1} - \lambda \boldsymbol{O}_A^{-1} - \boldsymbol{\Sigma} \right)}_{\boldsymbol{F}} \boldsymbol{e}_n \in \text{Convex Hull}(\{\boldsymbol{s}^{[i]} | s_n^{[i]} = \min_j s_n^{[j]}\}) \tag{8.53}$$

We'll now show that, if the sources are tightly scattered, definition 8.3.1, the answer is yes. First we do some work to identify the vector on the left-hand side as a point on the ellipse is maximally displaced in the direction $-\boldsymbol{e}_n$. We consider a vector on the ellipse, $\boldsymbol{v}_{\boldsymbol{x}}$, that is maximally displaced along some vector $\boldsymbol{x}$. To find $\boldsymbol{v}_{\boldsymbol{x}}$ we maximise $\boldsymbol{v}^T \boldsymbol{x}$ subject to the constraint that $\boldsymbol{v}_{\boldsymbol{x}}^T \boldsymbol{F} \boldsymbol{v}_{\boldsymbol{x}}$. Performing the resulting lagrange optimisation tells us that:

$$\boldsymbol{v}_{\boldsymbol{x}} = \frac{\boldsymbol{F}\boldsymbol{x}}{\sqrt{\boldsymbol{x}^T \boldsymbol{F} \boldsymbol{x}}} \tag{8.54}$$

Hence using $\boldsymbol{x} = -\boldsymbol{e}_n$ and the definition of $\boldsymbol{F}$, we get:

$$\boldsymbol{v}_{-\boldsymbol{e}_n} = \frac{1}{-\min_i[s_n^{[i]}]} \boldsymbol{F} \boldsymbol{e}_n \qquad [\boldsymbol{v}_{-\boldsymbol{e}_n}]_n = \min_i[s_n^{[i]}] \tag{8.55}$$

and we see that this point touches the bounding box $\min_i[s_n^i]$ along the nth axis. Therefore, if the ellipse is contained in the convex hull of the data, in particular if the ellipse kisses the ellipse at the minimising point along each axis, which it has to, then the gradient is zero.

So that's good, but we have to go to second order unfortunately. We can derive the horrendous second derivative in index notation, using Einstein summation convention, and δ_{ij} - the Kronecker delta:

$$\begin{aligned} \frac{\frac{1}{2}\partial^2 \mathcal{L}}{\partial W_{ij} \partial W_{kl}} &= -\lambda \delta_{ik} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{l\beta} (\boldsymbol{O}_A)_{\beta\gamma} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{\gamma j} + 2\lambda W_{i\alpha} W_{k\delta} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{\delta\alpha} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{l\beta} (\boldsymbol{O}_A)_{\beta\gamma} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{\gamma j} \\ &+ 2\lambda W_{i\alpha} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{\alpha\beta} (\boldsymbol{O}_A)_{\beta\gamma} W_{k\delta} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{\delta\gamma} (\boldsymbol{W}^T \boldsymbol{W})^{-1}_{lj} + \delta_{ik} (\Sigma_{lj} + \lambda (\boldsymbol{O}_A)^{-1}_{lj}) + \sum_n \delta_{ik} \frac{\partial \min_i[\boldsymbol{w}_n^T \boldsymbol{s}^{[i]}]}{\partial W_{ij}} \frac{\partial \min_i[\boldsymbol{w}_n^T \boldsymbol{s}^{[i]}]}{\partial W_{kl}} \end{aligned} \tag{8.56}$$

Now, this is a godawful mess. Fortunately, I derived it not you, and we don't have to conclude much from it. First notice that most terms have a δ_{ik}, this means they do not couple the perturbations of different neurons. There are only two terms that do. But, notice another aspect of both these two terms: they contain terms of the type $\boldsymbol{W}_{i\alpha}$ and $\boldsymbol{W}_{k\delta}$. We are going to evaluate this second derivative at $\boldsymbol{W} = \boldsymbol{D}$, and recall that the first d_S rows of $\boldsymbol{D}$ are a diagonal matrix, and the rest are zeros. So, if both i and k, the neuron indices, are below d_S, the loss will couple, but for all extra neurons, $i > d_S$, these two coupling terms will be zero, and they will contribute independently!

Therefore we split our evaluation of this quantity at the point $\boldsymbol{W} = \boldsymbol{D}$ into an analysis of the indices $i, k \leq d_s$, and the others. Let's begin with the more boring, $i, k \leq d_s$. Now there's no summation convention being applied any more, every quantity is just a scalar.

$$\frac{\frac{1}{2}\partial^2\mathcal{L}}{\partial W_{ij}\partial W_{kl}} = \delta_{ik}(\Sigma_{lj} + \lambda(\boldsymbol{O}_A)^{-1}_{lj}) + \lambda\delta_{ik}(d^*_l)^{-2}(\boldsymbol{O}_A)_{lj}(d^*_j)^{-2} + 2\lambda\delta_{lj}(d^*_j)^{-2}(d^*_i)^{-1}(\boldsymbol{O}_A)_{ik}(d^*_k)^{-1} + \sum_k \delta_{ik}[\boldsymbol{a}_k]_j[\boldsymbol{a}_k]_l \tag{8.57}$$

First notice that every term contains a kronecker delta over two of the indices and a positive semi-definite matrix over the other two[2]. This is all a bit of a pain to look at because we currently have a tensor with four indices. To turn it clean matrices we have to vectorise over the two matrix indices. Then each term is just the Kronecker product of two positive semi-definite matrices, since the Kronecker delta is just the identity matrix in index land. And, since the Kronecker product of two positive definite matrices is itself positive definite, this quantity will be too! Finally, we see that the 2nd derivative of the loss with respect to this portion of the linear map is positive definite, therefore all perturbations in these direction increase the loss.

Alternatively, we can take the case of a new neuron, $i = k = n$:

$$\frac{\frac{1}{2}\partial^2\mathcal{L}}{\partial W_{ni}\partial W_{nj}} = \underbrace{\Sigma_{ij} - \lambda(\boldsymbol{O}_A)^{-1}_{ij} + \lambda(d^*_i)^{-2}(\boldsymbol{O}_A)_{ij}(d^*_j)^{-2}}_{-F_{ij}} + \boldsymbol{a}_n\boldsymbol{a}_n^T \tag{8.58}$$

Again, $\boldsymbol{a}_n$ lives in the convex hull of the datapoints that minimise $\min_i(\boldsymbol{w}_n^T\boldsymbol{s}^{[i]})$, but we evaluated the derivative at the point $\boldsymbol{W} = \boldsymbol{D}$, i.e. where $\boldsymbol{w}_n = \boldsymbol{0}$. As such, all datapoints satisfy this minimia, and $\boldsymbol{a}_n$ is just a point that lives in the convex hull of the data.

To check whether this quantity will ever be zero we take the directional derivative along some direction $\boldsymbol{w}$:

$$\boldsymbol{w}^T\frac{\frac{1}{2}\partial^2\mathcal{L}}{\partial\boldsymbol{w}_n\partial\boldsymbol{w}_n}\boldsymbol{w} = -\boldsymbol{w}^T\boldsymbol{F}\boldsymbol{w} + (\boldsymbol{w}^T\boldsymbol{a}_n)^2 \tag{8.59}$$

So whether this quantity is negative, positive, or 0, all depends on the convex hull of the data, within which $\boldsymbol{a}_n$ lies. If we are able to choose $\boldsymbol{a}_n$ as the maximising point on the ellipse in the direction $\boldsymbol{w}$:

$$\boldsymbol{a}_n = \frac{\boldsymbol{F}\boldsymbol{w}}{\sqrt{\boldsymbol{w}^T\boldsymbol{F}\boldsymbol{w}}} \tag{8.60}$$

Then the derivative is clearly zero. If $\boldsymbol{a}_n$ is larger in the same direction the gradient is positive, and smaller it is negative.

Therefore the first sufficient scattering condition, theorem 5, which guarantees that $E \subseteq \text{Convex Hull}(\{\boldsymbol{s}^{[i]}\}_i)$, means that this quantity is nonnegative. Further, were this scattering condition not satisfied we would have found a direction which would reduce the loss, hence this condition is a necessary condition for the modular solution to be optimal.

And it is the second scattering condition that circumscribes the set of cases in which this gradient can be 0. The only allowed points of tangency between the convex hull and E are those that are extremal along a basis direction. This is saying two things. First, in all other directions the gradient is positive, and we cannot decrease the loss. Second, we can create a new neuron with $\boldsymbol{w} \propto \boldsymbol{e}_k \forall k = 1, ..., d_S$ and it won't change the loss, since the gradient is zero. Since this new neuron is a function of only a single source, this preserves the optimal representation's modularity. Were there more points of tangency, i.e. the second scattering condition was broken, there would be other perturbations that would leave the loss unchanged but introduce non-modularity, so we find that this condition is also necessary.

In sum, we find that, if both conditions hold, the modular representation is a local optima, while each condition is necessary. Hence the pair of conditions are the necessary and sufficient conditions for the optimal modular representation to be a local minima of the loss. Since the problem is a convex optimisation problem over the set of representational similarity matrices, $\boldsymbol{Q} = \boldsymbol{Z}^T\boldsymbol{Z}$, we know that, when these conditions hold, the optimal modular representation is not just a local minima but a global minima.

8.B.3 When is the Optimal Modular Solution *the* global minima

In section 8.2 we saw that out problem is a convex optimisation problem. We have found a globally minimal solution, our question is now: are there any other globally minimal solutions and if so are they also modular? We will show that under the tight scattering conditions all global minima are just permutations of the optimal modular solution.

[2] $\boldsymbol{\Sigma}$ is positive definite, each d^*_l is a positive number, $\boldsymbol{O}_A = \boldsymbol{A}^T\boldsymbol{A}$ so it and its inverse are positive definite too, and $\boldsymbol{a}_k\boldsymbol{a}_k^T$ is clearly positive semi-definite

All our representations are affine function of the demeaned data: $\boldsymbol{Z} = \boldsymbol{W}\bar{\boldsymbol{X}} + \boldsymbol{b}\boldsymbol{1}^T$, and we saw in section 8.2 that the readout weight loss was a strictly convex function of the dot product of the linear map relating the demeaned data to the representation: $\boldsymbol{W}^T\boldsymbol{W}$. This means that there is a uniquely optimal $\boldsymbol{W}^T\boldsymbol{W}$: any other globally optimal representation $\boldsymbol{Z}'$ must therefore have an input linear map that is a(n) (semi-)orthogonal transform of the optimal modular solution's, $\boldsymbol{W}^*$:

$$\boldsymbol{Z}' = \boldsymbol{O}\boldsymbol{W}^*\bar{\boldsymbol{X}} + \boldsymbol{b}'\boldsymbol{1}^T =\geq 0 \tag{8.61}$$

Further, in order to be optimal $\boldsymbol{Z}$ must have the same loss as $\boldsymbol{Z}_M$. We already know that they have the same readin and readout loss, since these are both fixed by the choice of readin transform $\boldsymbol{O}\boldsymbol{W}^*$, therefore they must also have the same activity loss. The activity loss of this new representation is:

$$\frac{1}{T}\operatorname{Tr}[\boldsymbol{Z}'^T\boldsymbol{Z}'] = \frac{1}{T}\operatorname{Tr}[\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^T\boldsymbol{W}^{*,T}\boldsymbol{W}^*] + ||\boldsymbol{b}'||^2 \tag{8.62}$$

Therefore we conclude that in order to have the same activity loss the new optimal solution's bias vector must have the same length as the original: $\boldsymbol{b}' = \boldsymbol{O}'\boldsymbol{b}^*$. Further, we know that in order for $\boldsymbol{Z}'$ to be an optimal representation $\boldsymbol{b}' = -\min_i[\boldsymbol{O}\boldsymbol{W}^*\boldsymbol{x}^{[i]}] = -\min_i[\boldsymbol{O}\bar{\boldsymbol{z}}_M^{[i]}]$, where $\bar{\boldsymbol{z}}_M^{[i]}$ is the demeaned optimal modular latent. Therefore, the puzzle becomes which rotations and reflections, $\boldsymbol{O}$, can be apply to $\bar{\boldsymbol{z}}_M^{[i]}$ such that the length of the minimal bias vector doesn't change? We'll now show that if the sources are tightly scattered $\boldsymbol{O}$ must be a permutation.

Since the sources obey the tight scattering assumptions, the optimal modular representation is tightly scattered with respect to a different ellipse, E_z:

$$E_z = \boldsymbol{D}E = \{\bar{\boldsymbol{z}}|\bar{\boldsymbol{z}} = \boldsymbol{D}\boldsymbol{y}, \quad \boldsymbol{y}^T\boldsymbol{F}^{-1}\boldsymbol{y} = 1\} = \{\bar{\boldsymbol{z}}|\bar{\boldsymbol{z}}^T\boldsymbol{D}^{-1}\boldsymbol{F}^{-1}\boldsymbol{D}^{-1}\bar{\boldsymbol{z}} = 1\} \tag{8.63}$$

Let's call the nth row of the orthogonal matrix $\boldsymbol{o}_n$, then we need to find $-\min_i[\boldsymbol{o}_n^T\bar{v}\bar{z}^{[i]}]$. Since the convex hull of the demeaned latents supersets E_z, we can upper bound this quantity $-\min_{\boldsymbol{y}\in E_z}[\boldsymbol{o}_n^T\boldsymbol{y}]$. The element $\boldsymbol{y}$ that achieves this can be calculated through lagrange optimisation to be:

$$-\frac{\boldsymbol{D}\boldsymbol{F}\boldsymbol{D}\boldsymbol{o}_n}{\sqrt{\boldsymbol{o}_n^T\boldsymbol{D}\boldsymbol{F}\boldsymbol{D}\boldsymbol{o}_n}}, \quad \text{hence} \quad -\min_{\boldsymbol{y}\in E_z}[\boldsymbol{o}_n^T\boldsymbol{y}] = \sqrt{\boldsymbol{o}_n^T\boldsymbol{D}\boldsymbol{F}\boldsymbol{D}\boldsymbol{o}_n} \tag{8.64}$$

Therefore we can calculate a lower bound on $||\boldsymbol{b}'||$, which can be simplified using the orthogonality of $\boldsymbol{O}$:

$$||\boldsymbol{b}'||^2 \geq \sum_n(-\min_{\boldsymbol{y}\in E_z}[\boldsymbol{o}_n^T\boldsymbol{y}])^2 = \sum_n \boldsymbol{o}_n^T\boldsymbol{D}\boldsymbol{F}\boldsymbol{D}\boldsymbol{o}_n = \operatorname{Tr}[\boldsymbol{D}^2\boldsymbol{F}] = \sum_j D_{jj}^2F_{jj} = \sum_j(d_j^*)^2(\min_i[\bar{s}_j^{[i]}])^2 = ||\boldsymbol{b}^*||^2 \tag{8.65}$$

where the penultimate equality follows from the fact that the diagonal elements of $\boldsymbol{F}$ are $(\min_i[\bar{s}_j^{[i]}])^2$, definition 8.3.1. So in fact this lower bound has to be achieved.

Further, we can now use the second tight scattering condition on the intersection of the boundary to constrain this orthogonal matrix. The convex hull of the data only touches the ellipse at points whose tangents are orthogonal to a basis direction. If $\boldsymbol{o}_n$ is not a basis direction then the inequality is strict: $(\min_i[\boldsymbol{o}_n^T\boldsymbol{z}^{[i]}])^2 > (\min_{\boldsymbol{y}\in E_Z}[\boldsymbol{o}_n^T\boldsymbol{y}])^2$, the bias vectors cannot be the same length, and the solution cannot be optimal. Therefore we derive that each $\boldsymbol{o}_n$ must be along a basis vector. The only way to achieve this is if the orthogonal matrix is a permutation.

8.B.4 Orthogonal Encoding Special Case

Finally, as a corollary we can derive our original linearly unmixed result, theorem 1. To do this we assume $\boldsymbol{x}^{[i]} = \boldsymbol{s}^{[i]}$. This occurs when $\boldsymbol{A} = I$, $\boldsymbol{O}_A = \boldsymbol{O}_A^{-1} = I$; then:

$$(d_j^*)^4 = \frac{\lambda}{\lambda + \langle(s_j^{[i]})^2\rangle_i + (\min_i s_j^{[i]})^2} \qquad (d_j^*)^{-4} = 1 + \frac{1}{\lambda}(\langle(s_j^{[i]})^2\rangle_i + (\min_i s_j^{[i]})^2) \tag{8.66}$$

Putting these into the definition of $\boldsymbol{F}$:

$$\boldsymbol{F} = \lambda\boldsymbol{D}^{-4} - \lambda - \langle\bar{\boldsymbol{s}}^{[i]}(\bar{\boldsymbol{s}}^{[i]})^T\rangle_i = \begin{bmatrix} (\min_i s_1^{[i]})^2 & -\langle s_1^{[i]}s_2^{[i]}\rangle_i & \dots \\ -\langle s_1^{[i]}s_2^{[i]}\rangle_i & (\min_i s_2^{[i]})^2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{8.67}$$

Exactly the $\boldsymbol{F}$ matrix from (William Dorrell, K. Hsu, et al., 2025).

8.C Identifiability of Neural Tuning Curves

We consider strictly convex optimisations over the set of representational dot-product similarity matrices with the constraint of nonnegative neural activities. In this setting there is a globally optimal similarity matrix, $\boldsymbol{Q}^*$. Let's imagine you measure a nonnegative representation that you suspect is optimal, $\boldsymbol{Z} \geq 0$, $\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{Q}^\star$. What has to be true about the single neuron properties of $\boldsymbol{Z}$ such that all optimal representations contain the same neurons?

In general, any other optimal representation must be an orthogonal transform of your original representation, $\boldsymbol{Z}' = \boldsymbol{O}\boldsymbol{Z}$. We consider constraints on $\boldsymbol{Z}$ such that, in order to preserve positivity, $\boldsymbol{O}$ must be a permutation matrix. In this case, the same single neuron responses appear in all representations, tieing them precisely to the optima.

We begin with a looser but simpler condition that has occured repeatedly in the literature. Then we show that we can significantly loosen this condition.

8.C.1 Sufficient Scattering

Following classic work on nonnegative matrix factorisation (Donoho and Stodden, 2003; Fu, Huang, and Sidiropoulos, 2018; Tatli and Erdogan, 2021a,b), define the cone:

$$\mathcal{C} = \{\boldsymbol{x} | \boldsymbol{x}^T\mathbf{1} \geq \sqrt{N-1}||\boldsymbol{x}||_2\} \tag{8.68}$$

Which has its corresponding dual cone[3]:

$$\mathcal{C}^* = \{\boldsymbol{x} | \boldsymbol{x}^T\mathbf{1} \geq ||\boldsymbol{x}||_2\} \tag{8.69}$$

Then our sufficient scattering condition is two conditions. We will use the notion of a cone of a matrix, all conic combinations of the columns of the matrix:

$$\text{cone}(\boldsymbol{X}) = \{\boldsymbol{x} | \boldsymbol{x} = \boldsymbol{X}\boldsymbol{a}, \forall \boldsymbol{a} \geq 0\} \tag{8.70}$$

1. First, a sufficient scattering condition that says that the datapoints, i.e. the rows of the matrix $\boldsymbol{Z}^*$, are spread around the positive orthant sufficiently:

$$\mathcal{C} \subseteq \text{cone}[\boldsymbol{Z}^T] \tag{8.71}$$

2. Second, there are few points at which the cone of the datapoints actually touches the edge of the cone $\mathcal{C}$. This can be formalised as:

$$\text{cone}\boldsymbol{Z}^{T^*} \cap \text{bd}\mathcal{C}^* = \{\lambda \boldsymbol{e}_k | \lambda \geq 0, k = 1, ..., N\} \tag{8.72}$$

   Where $\text{bd}(\mathcal{C}^*)$ is the boundary of $\mathcal{C}^*$.

Our claim is that, if these two sufficient scattering conditions are satisfied, then the single neuron response properties are unique.

Proof. It is a classic result from convex optimisation stuff, for two cones $\mathcal{K}_1$ and $\mathcal{K}_2$, if $\mathcal{K}_1 \subseteq \mathcal{K}_2$, then their duals satisfy the opposite relation: $\mathcal{K}_1^* \supseteq \mathcal{K}_2^*$. Since $\mathcal{C} \subseteq \text{cone}[\boldsymbol{Z}^T]$ we therefore have that $\mathcal{C}^* \supseteq \text{cone}[\boldsymbol{Z}^T]^*$

Now, imagine you have some other neural representation: $\boldsymbol{Z} = \boldsymbol{O}\boldsymbol{Z}^* \geq 0$ for some orthogonal matrix $\boldsymbol{O}$. Now, our first claim is about what has to be true about the rows of $\boldsymbol{O}$ such that $\boldsymbol{Z} \geq 0$. Call the nth row, $\boldsymbol{o}_n$, then, since $\boldsymbol{Z}^T\boldsymbol{o}_n \geq \mathbf{0}$:

$$\boldsymbol{o}_n \in \text{cone}(\boldsymbol{Z}^T)^* \subseteq \mathcal{C}^* \tag{8.73}$$

Now, membership in $\mathcal{C}^*$ means that $\boldsymbol{o}_n^T\mathbf{1} \geq ||\boldsymbol{o}_n||_2$. Further, we know that $||\boldsymbol{o}_n||_2 = 1$ because $\boldsymbol{O}$ is an orthogonal matrix, therefore $\boldsymbol{o}_n^T\mathbf{1} \geq 1$. In fact, we can go further and show this has to be an equality:

$$N = \mathbf{1}^T\mathbf{1} = \mathbf{1}^T\boldsymbol{O}^T\boldsymbol{O}\mathbf{1} = \sum_n (\boldsymbol{o}_n^T\mathbf{1})^2 \geq N \tag{8.74}$$

So, in order to satisfy the orthogonal matrix property, each $\boldsymbol{o}_n^T\mathbf{1} = 1$. This is great, because it means that $\boldsymbol{o}_n \in \text{bd}(\mathcal{C}^*)$. But now the vectors are trapped, we can use the second part of the scattering condition. Since each row $\boldsymbol{o}_n$ is both in $\text{bd}(\mathcal{C}^*)$ and $\text{cone}(\boldsymbol{Z}^T)^*$, it is in its intersection, and this is very prescribed by assumption:

$$\boldsymbol{o}_n \in \text{cone}\boldsymbol{Z}^{T^*} \cap \text{bd}\mathcal{C}^* = \{\lambda \boldsymbol{e}_k | \lambda \geq 0, k = 1, ..., N\} \tag{8.75}$$

Therefore, each row of $\boldsymbol{O}$ is a scaled version of the a basis element. But in order to be an orthogonal matrix, in fact this must just be a permutation matrix. □

[3] A dual cone for cone $\mathcal{C}$ is defined as $\mathcal{C}^* = \{\boldsymbol{y} | \boldsymbol{y}^T\boldsymbol{x} \geq 0 \forall \boldsymbol{x} \in \mathcal{C}\}$

8.C.2 A Family of Sufficient Conditions

Theorem 8 (Tight Scattering Implies Unique Neurons)**.** *If your neural data, $\boldsymbol{z}^{[i]}$, satisfies the following two tight scattering constraints with respect to a set $E = \{\boldsymbol{x} + \langle \boldsymbol{z}^{[i]} \rangle_i | \boldsymbol{x}^T \boldsymbol{F}^{-1} \boldsymbol{x} = 1\}$ for a positive definite matrix $\boldsymbol{F}$ with diagonals equal to $(\langle \boldsymbol{z}^{[i]} \rangle_i)^2$ then all orthogonal matrices such that $\boldsymbol{O}\boldsymbol{z}^{[i]} \geq 0$ are permutation matrices.*

- *Conv*$(\{\boldsymbol{z}_i^{[i]}\}) \supseteq E$
- *Conv*$(\{\boldsymbol{z}^{[i]}\}_i)^* \cap bd(E^*) = \{\lambda \boldsymbol{e}_k | \lambda \in \mathbb{R}, k = 1, ..., d_S\}$

Proof. The key constraint is that the transformed data must be positive: $\boldsymbol{Z}' = \boldsymbol{O}\boldsymbol{Z} \geq 0$. Consider a row of $\boldsymbol{O}$, $\boldsymbol{o}_n$. This has to satisfy:

$$\boldsymbol{o}_n^T \boldsymbol{z}^{[i]} \geq 0 \quad \forall i = 1...T \tag{8.76}$$

A necessary condition for this to be true is the same statement but for members of the ellipse E:

$$\boldsymbol{o}_n^T \boldsymbol{x} \geq 0 \quad \forall \boldsymbol{x} \in E \tag{8.77}$$

Given a putative transform vector $\boldsymbol{o}_n$, the member of E that will most tax its ability to preserve this defining inequality is the member of the ellipse E that has the largest projection along the $-\boldsymbol{o}_n$ direction. Using lagrange optimisation we can find that this is the vector:

$$\frac{-\boldsymbol{F}\boldsymbol{o}_n}{\sqrt{\boldsymbol{o}_n^T \boldsymbol{F} \boldsymbol{o}_n}} + \langle \boldsymbol{z}^{[i]} \rangle_i \tag{8.78}$$

Inserting this into eq. (8.76):

$$\boldsymbol{o}_n^T \left(\frac{-\boldsymbol{F}\boldsymbol{o}_n}{\sqrt{\boldsymbol{o}_n^T \boldsymbol{F} \boldsymbol{o}_n}} + \langle \boldsymbol{z}^{[i]} \rangle_i \right) = -\sqrt{\boldsymbol{o}_n^T \boldsymbol{F} \boldsymbol{o}_n} + \boldsymbol{o}_n^T \langle \boldsymbol{z}^{[i]} \rangle_i \geq 0 \tag{8.79}$$

If this inequality is satisfied then all those in eq. (8.76) are as well. Let's rewrite this once more as:

$$(\boldsymbol{o}_n^T \langle \boldsymbol{z}^{[i]} \rangle_i)^2 \geq \boldsymbol{o}_n^T \boldsymbol{F} \boldsymbol{o}_n \tag{8.80}$$

And this must hold concurrently for all the different rows of $\boldsymbol{O}$. Since $\boldsymbol{O}$ is an orthogonal matrix we know a lot about these rows, i.e. $\boldsymbol{o}_n^T \boldsymbol{o}_m = \delta_{nm}$. Let's put this information to use by considering the sum of these inequalities over the population:

$$\sum_n (\boldsymbol{o}_n^T \langle \boldsymbol{z}^{[i]} \rangle_i)^2 = \langle \boldsymbol{z}^{[i]} \rangle_i^T \boldsymbol{O}^T \boldsymbol{O} \langle \boldsymbol{z}^{[i]} \rangle_i = ||\langle \boldsymbol{z}^{[i]} \rangle_i||_2^2 \geq \sum_n \boldsymbol{o}_n^T \boldsymbol{F} \boldsymbol{o}_n = \mathrm{Tr}[\boldsymbol{O}^T \boldsymbol{F} \boldsymbol{O}] = \mathrm{Tr}[\boldsymbol{F}] = ||\langle \boldsymbol{z}^{[i]} \rangle_i||_2^2 \tag{8.81}$$

Where the last inequality follows from the fact that the diagonal elements of $\boldsymbol{F}$ are $(\langle \boldsymbol{z}^{[i]} \rangle_i)^2$. Further, since the sums are equal, each of the elements must be equal. Hence, we see that if the first sufficient scattering condition is satisfied, then this inequality constraint must be an equality:

$$\boldsymbol{o}_n^T \underbrace{\left(\frac{-\boldsymbol{F}\boldsymbol{o}_n}{\sqrt{\boldsymbol{o}_n^T \boldsymbol{F} \boldsymbol{o}_n}} + \langle \boldsymbol{z}^{[i]} \rangle_i \right)}_{\in E_Z} = 0 \tag{8.82}$$

Now we use the second scattering condition. The positivity constraint, $\boldsymbol{o}_n^T \boldsymbol{z}^{[i]} \geq 0 \forall i$, is the defining property of membership of the dual cone of the data: $\boldsymbol{o}_n \in \mathrm{Conv}(\{\boldsymbol{z}_i\}_i)^*$. Further, since $\boldsymbol{o}_n^T \boldsymbol{x} \geq 0 \forall \boldsymbol{x} \in E$, it is also in the ellipse's dual cone $\boldsymbol{o}_n \in \mathrm{Conv}(E)^*$. Finally, not only is $\boldsymbol{o}_n$ in the dual cone of the convex hull of E, it is on the boundary, because, as shown in the equation above, $\boldsymbol{o}^n \boldsymbol{x} = 0$ for a point $\boldsymbol{x} \in E$. Therefore:

$$\boldsymbol{o}_n \in \mathrm{Conv}(\{\boldsymbol{z}_i\}_i)^* \cap \mathrm{bd}(E^*) = \{\lambda \boldsymbol{e}_k | \lambda \in \mathbb{R}, k = 1, ..., d_S\} \tag{8.83}$$

Where the last equality is the second scattering condition. Hence we have our final result. In order to have the same dot product structure and preserve positivity each $\boldsymbol{o}_n$ must align with a basis direction, and since it is unit norm, it must be a basis direction. So each new neuron $\boldsymbol{z}_n'^{[i]} = \boldsymbol{o}_n \boldsymbol{z}^{[i]} = z_k^{[i]}$: all single unit tuning properties are preserved. □

8.C.3 Partial Identifiable Populations

In practice, it might be inconvenient to expect every neuron in a population to be identifiable - perhaps some set of neurons are together encoding one variable and can be easily rotated amongst themselves, but not mixed with another population. We study a setting in which such statements can also be proven. We consider an optimisation problem that is strictly convex over the de-meaned representational similarity matrices:

$$\bar{\boldsymbol{Q}} = \bar{\boldsymbol{Z}}^T \bar{\boldsymbol{Z}}, \quad \bar{\boldsymbol{Z}} = \boldsymbol{Z} - \boldsymbol{Z}\mathbf{1}\mathbf{1}^{\mathbf{T}} \tag{8.84}$$

In this setting, all optimal representations have to be rotations of the demeaned sources plus some bias. Assuming positivity and activity regularisation, you can further prove that to be optimal the minimal bias of the original and rotated sources have to have equal length (see previous section). We now show that, if two sets of sources are range-independent, no representation that mixes them will be optimal.

Let's consider a matrix, $\boldsymbol{O}$, with orthogonal columns that transforms the de-meaned representation while preserving its dot product structure:

$$\tilde{\boldsymbol{z}}_i = \boldsymbol{O}\bar{\boldsymbol{z}}_i \qquad \tilde{\boldsymbol{z}}_j^T \tilde{\boldsymbol{z}}_i = \bar{\boldsymbol{z}}_j \boldsymbol{O}^T \boldsymbol{O} \bar{\boldsymbol{z}}_i = \bar{\boldsymbol{z}}_j \bar{\boldsymbol{z}}_i \tag{8.85}$$

Let's say that:

$$\bar{\boldsymbol{z}_i} = \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{y}_i \end{bmatrix} \tag{8.86}$$

For two range independent variables $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$. Consider an orthogonal matrix that mixes the representation: $\boldsymbol{O}$. Then consider breaking the mixed parts into two separate populations of neurons:

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{O}_x & \boldsymbol{O}_y \end{bmatrix} \qquad \boldsymbol{O}' = \begin{bmatrix} \boldsymbol{O}_x & \mathbf{0} \\ \mathbf{0} & \boldsymbol{O}_y \end{bmatrix} \tag{8.87}$$

A representation made from $\boldsymbol{O}'$ still has the optimal dot-product similarity, because the columns of $\boldsymbol{O}'$ remain orthogonal. However, we will now show that the representation made from $\boldsymbol{O}'$ must have a shorter bias vector, and therefore the mixed representation cannot be an optimal one. Let's calculate the two bias vectors:

$$||\boldsymbol{b}||^2 = || - \min_i [\boldsymbol{O}_x \boldsymbol{x}_i + \boldsymbol{O}_y \boldsymbol{y}_i]||^2 = || \min_i [\boldsymbol{O}_x \boldsymbol{x}_i] + \min_i [\boldsymbol{O}_y \boldsymbol{y}_i]||^2 \geq || \min_i [\boldsymbol{O}_x \boldsymbol{x}_i]||^2 + || \min_i [\boldsymbol{O}_y \boldsymbol{y}_i]||^2 \tag{8.88}$$

And the final term is exactly the length of the bias required for the representation constructed from $\boldsymbol{O}'$. The equality only occurs if $\min_i [\boldsymbol{O}_x \boldsymbol{x}_i]^T \min_i [\boldsymbol{O}_y \boldsymbol{y}_i] = 0$. Since the variables are mean-zero, this only happens if $\boldsymbol{O}$ already kept the representation separate. Therefore, any time we think we find an optimal solution, it can be improved by keeping range-independent variables in different neurons.

8.D ON-OFF Coding

We consider a representation, $\boldsymbol{z}^{[i]}$ that is nonnegative and from which a single stimulus, $x^{[i]}$, can be decoded using an affine readout:

$$\boldsymbol{R}\boldsymbol{z}^{[i]} + \boldsymbol{r} = x^{[i]} \tag{8.89}$$

Subject to this encoding constraint and the nonnegativity of the representation we minimise an energy loss:

$$\mathcal{L} = \langle ||\boldsymbol{z}^{[i]}||^2 \rangle_i + \lambda ||\boldsymbol{R}||_F^2 \tag{8.90}$$

In section 8.2 we showed that this problem is convex. We will use the KKT conditions to find the optimal representations $\boldsymbol{Z}$ ($\boldsymbol{Z}$ denotes the stacked representation matrix, $[\boldsymbol{Z}]_{:,i} = \boldsymbol{z}^{[i]}$ and similarly $[\boldsymbol{x}]_i = x^{[i]}$). We will find that the representation either comprises both an ON and an OFF channel, or just a single channel, and we will show that sparsity delimits these two scenarios. Using convexity, these must correspond to a global optima up to an orthogonal matrix, and, for these simple representations, it's easy to use the results in section 8.4 to show that ON-OFF coding cannot be rotated.

8.D.1 KKT Conditions lead to ON-OFF coding

The min-norm readout vector, $\boldsymbol{R}$, is the pseudoinverse that maps the de-meaned representation, $\bar{\boldsymbol{Z}}$, to the de-meaned data, $\bar{\boldsymbol{x}}$. Using this, we can write the loss as:

$$\mathcal{L} = \frac{1}{T}\operatorname{Tr}[\boldsymbol{Z}^T\boldsymbol{Z}] + \lambda \operatorname{Tr}[\bar{\boldsymbol{Z}}^\dagger \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T(\bar{\boldsymbol{Z}}^\dagger)^T] = \frac{1}{T}\operatorname{Tr}[\boldsymbol{Z}^T\boldsymbol{Z}] + \lambda \bar{\boldsymbol{x}}^T(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{Z}}^\dagger\bar{\boldsymbol{x}} \tag{8.91}$$

This has to be minimised subject to non-negativity, so we construct the following lagrangian:

$$\mathcal{L} = \frac{1}{T}\operatorname{Tr}[\boldsymbol{Z}^T\boldsymbol{Z}] + \lambda \bar{\boldsymbol{x}}^T(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{Z}}^\dagger\bar{\boldsymbol{x}} - \operatorname{Tr}[\boldsymbol{P}^T\boldsymbol{Z}] \tag{8.92}$$

We take the derivative with respect to $\boldsymbol{Z}$ and, using the expression for the derivative of the pseudoinverse of constant rank (Golub and Pereyra, 1973), and the fact that $\boldsymbol{Z}^\dagger\bar{\boldsymbol{x}}$ must always be rank 1 to solve the task, we find that:

$$\frac{1}{T}\boldsymbol{Z} - \lambda \boldsymbol{Z}\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T = \frac{1}{2}\boldsymbol{P} \tag{8.93}$$

And further, if $Z_{ij} > 0$ then $P_{ij} = 0$, or if $P_{ij} > 0$ then $Z_{ij} = 0$. If the former case:

$$Z_{ij} = T\lambda[\boldsymbol{Z}\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T]_{ij} \tag{8.94}$$

Else it is zero. Since $P_{ij} \geq 0$ in these cases we know that $\boldsymbol{Z}\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T \leq 0$. Hence, this can all be summarised by the following equation:

$$Z_{ij} = T\lambda[\boldsymbol{Z}\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T]_+ \tag{8.95}$$

Where $[]_+$ denotes the elementwise operation $[x]_+ = \max(x, 0)$.

We can study the firing of a single neuron, $\boldsymbol{z}_n \in \mathbb{R}^T$:

$$\boldsymbol{z}_n = T\lambda[\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\boldsymbol{z}_n]_+ \tag{8.96}$$

Noticing that the same vector appears twice: $\boldsymbol{a} = \bar{\boldsymbol{Z}}\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{z}}$, we can rewrite much more simply:

$$\boldsymbol{z}_n = T\lambda[\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{z}_n]_+ \tag{8.97}$$

We can break $\boldsymbol{a}$ into two parts, its positive and negative components, $\boldsymbol{a}_+ = [\boldsymbol{a}]_+$, $\boldsymbol{a}_- = [-\boldsymbol{a}]_+$, then this equation already tells us that in the optimal population there are only two types of neural responses, governed by the sign of $\boldsymbol{a}^T\boldsymbol{z}_n$:

$$\boldsymbol{z}_n = \begin{cases} T\lambda\boldsymbol{a}^T\boldsymbol{z}_n\boldsymbol{a}_+ & \text{if} \quad \boldsymbol{a}^T\boldsymbol{z}_n > 0 \\ T\lambda|\boldsymbol{a}^T\boldsymbol{z}_n|\boldsymbol{a}_- & \text{if} \quad \boldsymbol{a}^T\boldsymbol{z}_n < 0 \end{cases} \tag{8.98}$$

Now we derive these responses using the definition of $\boldsymbol{a}$: $\boldsymbol{a} = \bar{\boldsymbol{Z}}^\dagger(\bar{\boldsymbol{Z}}^\dagger)^T\bar{\boldsymbol{x}}$. Since $\bar{\boldsymbol{x}}$ is in the span of $\bar{\boldsymbol{Z}}$:

$$\bar{\boldsymbol{Z}}^T\bar{\boldsymbol{Z}}\boldsymbol{a} = \bar{\boldsymbol{x}} \tag{8.99}$$

Now, all neurons belong to these two firing patterns. Call the set of positive neuron indices S_+ then define the sum of all positive weightings $\alpha_+ = \sum_{n\in S_+}|\boldsymbol{a}^T\boldsymbol{z}_n|$, and the same for negative. Then:

$$\bar{\boldsymbol{Z}}^T\bar{\boldsymbol{Z}}\boldsymbol{a} = \alpha_+\bar{\boldsymbol{z}}_+\bar{\boldsymbol{z}}_+^T\boldsymbol{a} + \alpha_-\bar{\boldsymbol{z}}_-\bar{\boldsymbol{z}}_-^T\boldsymbol{a} \tag{8.100}$$

Hence:

$$T^2\lambda^2(\alpha_+\bar{\boldsymbol{a}}_+\bar{\boldsymbol{a}}_+^T\boldsymbol{a} + \alpha_-\bar{\boldsymbol{a}}_-\bar{\boldsymbol{a}}_-^T\boldsymbol{a}) = \bar{\boldsymbol{x}} \tag{8.101}$$

Where we've used the de-meaned variables $\bar{\boldsymbol{a}}_\pm$. Expanding these in terms of their means, $\mu_\pm$: $\bar{\boldsymbol{a}}_\pm = \boldsymbol{a}_\pm + \mu_\pm\mathbf{1}$ and writing the mean of $\boldsymbol{a}$ as μ_a, we find that, with some rearrangement:

$$T^2\lambda^2\left((\frac{1}{T\lambda} - T\mu_a)\boldsymbol{a}_+ - (\frac{1}{T\lambda} + T\mu_a)\boldsymbol{a}_-\right) = \bar{\boldsymbol{x}} - T^2\lambda^2(\mu_a T(\mu_+ + \mu_-) + \frac{\mu_+ - \mu_-}{T\lambda})\mathbf{1} = \bar{\boldsymbol{x}} + b\mathbf{1} \tag{8.102}$$

Since $\boldsymbol{a}_+$ and $\boldsymbol{a}_-$ are never concurrently non-zero, this tells us that they each code for a portion of $\bar{\boldsymbol{x}} + b\mathbf{1}$, $\boldsymbol{a}_+$ the positive, $\boldsymbol{a}_-$ the negative. We could wade further through this mess, but instead we can skip to a 2 neuron representation with exactly this form and optimise the remaining parameters directly:

$$\boldsymbol{z}^{[i]} = \begin{bmatrix} \alpha_+[x^{[i]} - b]_+ \\ \alpha_-[-(x^{[i]} - b)]_+ \end{bmatrix} \tag{8.103}$$

Then we can calculate the loss:

$$\mathcal{L} = \langle||\boldsymbol{z}^{[i]}||_2^2\rangle_i + \lambda||\boldsymbol{R}||_F^2 = \alpha_+^2\langle[x^{[i]} - b]_+^2\rangle_i + \alpha_-^2\langle[-(x^{[i]} - b)]_+^2\rangle_i + \lambda(\frac{1}{\alpha_+^2} + \frac{1}{\alpha_-^2}) \tag{8.104}$$

We will assume each b lies within the range of x (i.e. we are using an ON-OFF code) and take the derivative with respect to each of the α to get:

$$\alpha_+^4 = \frac{\lambda}{\langle[x^{[i]} - b]_+^2\rangle_i} \qquad \alpha_-^4 = \frac{\lambda}{\langle[-(x^{[i]} - b)]_+^2\rangle_i} \tag{8.105}$$

And with respect to the bias we end up with the implicit equation:

$$\frac{\int_b^\infty (x-b)dp(x)}{\int_b^\infty (x-b)^2dp(x)} = \frac{\int_{-\infty}^b (b-x)dp(x)}{\int_{-\infty}^b (b-x)^2dp(x)} \tag{8.106}$$

So, in general, solve this implicit equation for b (even numerically), then calculate α_+ and α_- and you have your representation. A simpler solution is that if $p(x)$ is symmetric you can see that $b = 0$ is a solution.

8.D.2 Direct vs. ON-OFF Coding

The previous result assumed that we had an ON and an OFF neuron each with non-zero firing. The other alternative is a single channel, for which we have two choices, ON or OFF:

$$\text{either: } z^{[i]} = x^{[i]} - \min_i x^{[i]} \text{ or: } z^{[i]} = -x^{[i]} - \max_i x^{[i]} \tag{8.107}$$

We only have to consider the optimal one, which will be the lower energy option. This will be the ON channel if:

$$\langle(x^{[i]} - \min_i x^{[i]})^2\rangle_i \leq \langle(-x^{[i]} - \max_i x^{[i]})^2\rangle_i \tag{8.108}$$

and the OFF channel otherwise.

For simplicity let's assume the ON channel is the preferred single neuron response (all the arguments can be switched to the OFF if needed). Let's calculate what has to be true for the one neuron solution to be stable to perturbations into two neuron coding.

To do that we compare the one neuron loss for the optimal value of $\alpha_+ = \sqrt{\frac{\lambda}{\langle(x^{[i]} - \min_i x^{[i]})^2\rangle_i}}$:

$$\mathcal{L}_+ = 2\sqrt{\lambda\langle(x^{[i]} - \min_i x^{[i]})^2\rangle_i} \tag{8.109}$$

To one where we shift the bias the tiny-iest amount positive to create an ON and OFF neuron. Since the dataset is finite, we can choose this tiny bias to sit between the minimal x value and the next smallest. Then the OFF neuron will only activate for the minimal x. Let's denote with S the probability of the minimal x, then the loss:

$$\mathcal{L} = \alpha_+^2\langle[x^{[i]} - \min_i x^{[i]} - b]_+^2\rangle_i + \alpha_-^2 b^2 S + \lambda(\frac{1}{\alpha_+^2} + \frac{1}{\alpha_-^2}) \tag{8.110}$$

We can study the second moment:

$$\langle[x^{[i]} - \min_i x^{[i]} - b]_+^2\rangle_i = \langle(x^{[i]} - \min_i x^{[i]} - b)^2\rangle_i - b^2S = \langle(x^{[i]} - \min_i x^{[i]})^2\rangle_i - 2b\langle x^{[i]} - \min_i x^{[i]}\rangle_i + b^2(1 - S^2) \tag{8.111}$$

And then, using this expression, calculate the optimal α_+ and α_- and find, to first order in b:

$$\mathcal{L} = 2\sqrt{\lambda b^2 S} + 2\sqrt{\lambda \langle (x^{[i]} - \min_i x^{[i]})^2 \rangle_i - 2b\langle x^{[i]} - \min_i x^{[i]} \rangle_i} \tag{8.112}$$

Comparing this to the previous loss and series expanding in b:

$$\frac{\mathcal{L} - \mathcal{L}_+}{2\sqrt{\lambda}} = b\sqrt{S} - b\frac{\langle x^{[i]} - \min_i x^{[i]} \rangle_i}{\sqrt{\langle (x^{[i]} - \min_i x^{[i]})^2 \rangle_i}} \tag{8.113}$$

Hence, dual ON-OFF channel coding is better if:

$$S < \frac{\langle x^{[i]} - \min_i x^{[i]} \rangle_i^2}{\langle (x^{[i]} - \min_i x^{[i]})^2 \rangle_i} \tag{8.114}$$

else, when the representation is sparse enough, single channel coding is better.

Chapter 9

Mixed Selectivity Discussion

In this part of the thesis we sought to understand when and why neurons deep in cortex might appear to modularly code for meaningful variables, while at other times mixing their encodings together. This builds into a broad, and in my opinion very deep, question about how we should expect a complex computing device like the brain to divide its computation in to chunks. If we are to understand the brain we have to identify a set of meaningful chunks, and these theories are steps towards defining the word meaningful in this context. That said, they are very limited steps, as we now outline.

9.1 Limitations

At a finescale, the theory has a lot of clear shortcomings. Some of our identifiability conditions, for example for norms other than L2 section 5.A.4, or multidimensional sources section 5.A.5 are sufficient but not necessary, guaranteeing that, even within our current problem settigs, there are some types of modularity we are not catching. Further our notion of modularity is very binary: if even a single neuron encodes two variables the representation is classed as mixed selective. A more analogue notion of modularity might be more useful. Finally, because we have to assume perfect reconstruction error, even a point with infinitesimally low probability has to be correctly encoded, potentially changing the structure of the optimal solution. If we included the reconstruction error as a cost rather than a constraint, we could gracefully trade-off energy costs for reconstruction error. Empircally this is what appears to happen (William Dorrell, K. Hsu, et al., 2025), but theory on the case of non-perfect reconstruction would be more powerful.

At a slightly more fundamental level, our theory's reliance on linearity is heavily constraining. At the moment this seems theoretically insurmountable, and the likely avenue for progress seems to be the standard one: guessing the right variables such that the problem of interest becomes linear enough. Despite this pessimism, there has been a lot of recent work on nonlinear ICA (Buchholz, Besserve, and Schölkopf, 2022; Horan, Richardson, and Weiss, 2021; Hyvarinen, Khemakhem, and Morioka, 2023; Khemakhem et al., 2020; Kivva et al., 2022; Klindt et al., 2021; Lachapelle and Lacoste-Julien, 2022; Lachapelle, Rodriguez, et al., 2022; Moran et al., 2022; Sprekeler, Zito, and Wiskott, 2014; X. Yang et al., 2022). These works develop identifiability results that reflect properties of the nonlinear map between data and representation, and similar approaches might hold promise for us. Finally, we sound two positive notes regarding the linearity of our results. First, empirically, the results we derived seem to generalise well to nonlinear networks (William Dorrell, K. Hsu, et al., 2025; J. C. Whittington, Will Dorrell, et al., 2023), so much so that collaborators and I used insights from this theory to develop a state-of-the-art disentangling method (K. Hsu, William Dorrell, et al., 2023; K. Hsu, Hamid, et al., 2024). Second, the analysis presented in chapter 8 demonstrates an approaches to studying nonlinear, but linearly decodable representations of data, such as a the feature map in a kernel regression problem, or the last layer in a neural network. This is a significant relaxation that we have already found useful in unpublished work.

Finally, most of the work in this part was as computationally boring as the original efficient coding hypothesis: these representations weren't doing anything. We were simply optimising various reformattings of the data without requiring the representation to be the substrate of any computation, antithetical to this thesis' stated goals! We made some small progress on this question studying linear recurrent neural networks, chapter 6, however this proved surprisingly difficult. Yet, this is the natural setting for neuroscience. A dream result would be an identifiability result for computational motifs: what has to be true such that an RNN learns a modular computation. This could be used to reason about the modularity observed empircally in RNNs (Driscoll, Shenoy, and Sussillo, 2022; G. R. Yang et al., 2019). Results similar to these have been derived in a linear hypernetwork setting (Schug et al., 2024). Sharpening our tools to answer such a problem does not seem impossible, a topic we return to in chapter 13.

One last limitation is a lack of experimental verification. We managed to postdict some data, chapter 7,but

every theory that ever sees the light of day can do that. Ideally we would design a pair of rooms with different reward distributions one of which should modularise, the other mix, and measure the resulting encodings. We are running experiments like this in the Behrens lab, and while there are a lot of caveats, as always, this will be very interesting datapoint.

9.2 Mixed Selectivity in Machines

As discussed above, our work has contributed to a state-of-the-art disentangling method (K. Hsu, William Dorrell, et al., 2023; K. Hsu, Hamid, et al., 2024) that bears similarities to the biological constraints covered here. In this respect it is interesting that, unlike more classical problems like ICA (Comon, 1994), the type of independence being sought by these methods depends much more on the range. Roth et al. (2023) discuss simiar ideas before concluding that this is often a more natural form of independence. They give an example of images composed from two animals, a cow or a camel, on two backgrounds, either a desert of a field. These variables are not independent: cows like grass, camels like deserts. But just one example of a cow on sand and a camel on grass is enough to make them range-independent. If the dataset contained such an example methods like ours would disentangle them. It is intriguing the form of independence that some disentangling methods are converging to is precisely the one that our biologically inspired constraints cares about.

Further, a lot of recent work has focused on studying the mixed-selectivity of representations inside artificial neural networks (Elhage et al., 2022). Findings have shown that many neurons are meaningfully mixed-selective, making it hard to understand their role. Dictionary learning has been used to disentangle the factors of variation in these models, with surprising levels of success. Bricken et al. (2023) were able to extract meaningful variable, like 'the golden gate bridge'. More impressively, these variables were causal. By artificially activating these discovered encodings during the forward pass of the model, the large language model they were studying began talking about the golden gate bridge incessently, and often in very creative ways. In the best example, the LLM and the user were playing noughts and crosses when the golden gate bridge concept was activated. It's tough to bring up the golden gate bridge in a game of noughts of crosses, the model managed this by using a golden gate emoji rather than a nought or cross to make its play! Given these successes of dictionary learning, we are prompted to consider a pleasingly recursive use case of the theory we have developed: understanding the methods we use to disentangle neural networks, a topic I am currently working on.

9.3 Mixed Selectivity in Brains

As recording equipment has improved, mixed selectivity is becoming a relatively common finding in neuroscience Tye et al., 2024. Why is the nervous system built this way?

Way back in chapter 2 we bumped into our first example of mixed selectivity in this thesis: conjunctive heading-velocity neurons in the fly ring attractor. These neurons respond to a conjunction of a given heading and velocity; they are what we might call *structured mixed selectivity*. They implement a meaningful computation, and from examining patterns in such mixed selectivity - i.e. the presence of all combinations of only two variables, recurrently connected to a population of pure selective heading cells (Hulse and Jayaraman, 2020), or grid cells in the analogous 2D problem (Vollan et al., 2025) - we might hope to infer the computation occurring in the circuit from these observations. I strongly approve of these kinds of analyses of mixed selectivity. That said, these neurons, precisely due to their structured nature, are rarely referred to as mixed selective. Mixed selectivity presumes confusion, we're not confused about these neurons, they have a clear job to do!

The work presented here has pointed to two further reasons neurons might be mixed selective, chapter 7. First, if the encoded variables are insufficiently independent, energy efficiency arguments have told us that they might be usefully encoded in a mixed representation. Our theory has pointed to linearly mixed selective neurons, but actually the results empirically generalise to nonlinear mixed selectivity. Second, we might be inferring mixed selectivity in what are actually modular populations, due to correlations between the variable we analyse with respect to and those actually encoded in the population. This points to a broader problem with the framework of mixed-selectivity: it's variable dependence. Given a representation I can always find a set of variables that make its encoding mixed, or modular. This points to the importance of carefully thinking through the variables you use. As far as I can tell, this thorny issue has not been cleanly thought through in general. Thankfully in behavioural tasks we get to choose the variables, and if the animal is performing the task sufficiently well we have to presume it understands the variables we have designed. This is a much tougher question in more innate domains, like vision or motor control, where a large and unsolved part of the problem is finding the meaningful latent variables.

A fourth example of mixed selectivity arising is discused by Masset et al. (2022). In sampling-based samplers mixed selectivity emerges as a property that allows for speedier computations than otherwise.

However, the prevailing discussion of mixed selectivity focuses on a different notion entirely: (random) nonlinear mixed selectivity for linear decodabilty of flexible readouts (Rigotti et al., 2013; Tye et al., 2024). These theories argue that mixed selectivity is forming a flexible nonlinear representation from which arbitrary functions can be linearly readout, like a kernel regression algorithm. I can find these kernel regression theories compelling, specifically when applied to neural circuits that really look anatomically like large expansive projections from which flexible linear decoders are learnt, such as the cerebellum and related circuits (see part IV). However, these theories are regularly applied to PFC. I find this preposterous! The PFC is the home of our internal models at increasing levels of abstractions, the locus of our planning and active working. A sufficiently flexible kernel machine can indeed perform these tasks, just like a one-hidden layer neural network is a universal function approximator. However, this strikes me as the most uncortical implementation imaginable.

On a more structural point, any misunderstanding of the algorithms being performed by a brain area can appear as mixed selectivity, making it a worryingly easy fall-back explanation. If you did not know the central complex of insects was involved in heading direction you might analyse it for encodings of objects in the visual field. In a small dataset there will certainly be correlations, since an object far away somewhat approximates your heading direction. Given enough data collected from many vantage points these correlations will disappear, but then you can always analyse for tuning to the conjunction of visual object and hunger, or some other way of dividing the variables, and suddenly you need more data again to discount this possibility.

As such, while I find this a convincing neural theory in some cases, I am unconvinced this is often a useful way to think about cortical representations. There are probably cases it is useful, in the same way that it might be useful to think about conjunctive heading-velocity neurons as a large expansive population from which you can linearly decode your next heading direction. But I think it should be a theory of last resort: only when the neuroscientist has been thoroughly battered by the incomprehensibility of the measured neural tunings may they reach for this theoretical cheatcode.

9.4 Conclusion

We have presented a normative theory of mixed selectivity and modularity in the brain that predicts patterns of modularisiation in some otherwise puzzling neural data, and holds promise for similar approaches in nonlinear disentangling. We then extended these approaches to study modularisation in linear recurrent networks. Finally, we introduced a set of convex representational optimisation problems, and used this framework to generalise our theoretical findings further, to the case of linearly mixed sources. Doing this gave us a series of identifiability results, either for neural tuning curves, or a necessary and sufficient condition for matrix factorsiation problems. In the next part we return to the core thread of this thesis, and use an efficient computing approach, combined with our newfound understanding of modularity, to tackle structured working memory representations in the prefrontal cortex.

Part III

Structured Prefrontal Working Memory Representations

Welcome to the third act of this thesis, in many ways the finale (you'll soon realise that act IV is more an encore than it is a cohesive member of this thesis). To situate ourselves, we began this thesis arguing that we needed normative theory for representations performing computations rather than communicating, which we immediately developed in part I into a normative theory of grid cells. Then, in part II, we considered when we should normatively expect modular structure to arise in neural activity. On first blush, this appears a detour (indeed, it was definitely longer than absolutely necessary) but these explorations were prompted by the work we will now discuss. Here, we consider another cortical computation and attempt to frame a corresponding efficient computing theory. In particular, we move to the prefrontal cortex and construct a theory of structured working memory.

The prefrontal cortex is often labelled the executive region, capable of storing the current state of the world, controlling attention, and constructing plans. As we will review, decades of electrophysiology work has begun to pin down the neural basis of these abilities. The personal history is that this normative theory was constructed to think about the recordings of Dr. Mohamady El-Gaby (El-Gaby et al., 2024). He recorded medial frontal activity in mice trained to learn then visit a sequence of four rewarded port locations in a maze containing nine. The measured neural activity was intricate, showing clear evidence of both working memory (what ports are currently rewarded) and structure (in what order are the ports rewarded). I spent a long time puzzling about this representation and these normative theories are the result. Similarly, James Whittington constructed neural network models of this task, and others like it, and showed that some of the measured neural responses could emerge from simple neural networks (J. C. Whittington, William Dorrell, et al., 2025). My contributions to both these cited papers were very minor, so, though they guide this work, here we focus on presenting a cohesive summary of the normative theorising.

This problem is a perfect second development for efficient computing type approaches. As in the path-integrating work, part I, we construct an optimal neural representation that tracks your position within a structure, such as a simple sequence. However, unlike the grid cell code, the representation must subserve working memory - it must encode not just where you are, but what you should expect to find there. The optimal representations we find match a variety of aspects of the neural recordings, and their neural network models. In particular, we make repeated use of the developments on modular coding to understand observed patterns of modularisation in these prefrontal models. The causality in fact goes in reverse, we saw these puzzling prefrontal structures, and developed the modularity results to understand them. Further, this theorising makes some very clear predictions, some of which are likely to be tested soon! (see chapter 13)

As such, this theory matches many puzzling details of the measured representations. Yet, as you will see, and despite much agonising, there are aspects that do not match. First, the single neuron predictions are often wrong. Second, there is a whole layer of structure missing from our theoretical representations. It seems likely these two are linked and that a more complete theory could capture them. More optimistically, this outlines the interesting role of normative theorising as null model: since we are unable to fit these aspects of the data using our simple model, they must reflect some underappreciated aspect of the problem that the measured representation is solving. I look forward to learning what this aspect is before I die.

Chapter 10

A NORMATIVE THEORY OF STRUCTURED PREFRONTAL WORKING MEMORY REPRESENTATIONS

The prefrontal cortex is hypothesised to form the mind's workspace. Among its putative functions, two ideas with significant neural evidence are working memory encoding (Funahashi, Chafee, and Goldman-Rakic, 1993; Fuster and Alexander, 1971; Kubota and Niki, 1971), and the representation of structural schemas (Baldassano, Hasson, and Norman, 2018). Building on observations going back decades (Mushiake et al., 2006; Ohbayashi, Ohki, and Miyashita, 2003; Shima, Isoda, et al., 2007), recent work has examined the prefrontal representation of tasks that combine both working memory and structural components (Basu et al., 2021; J. Chen et al., 2024; El-Gaby et al., 2024; Panichello and Buschman, 2021; Tian et al., 2024; Y. Xie et al., 2022), such as a sequence working memory task: you see a sequence of stimuli ABC and have to recall them (working memory) in the order presented (schema) (Y. Xie et al., 2022) or perhaps reversed (Tian et al., 2024). These representations show intriguing structure, with memories encoded in different subspaces within the neural activity, while schematic features are encoded in the relationship between the subspaces. Modelling work has shown that simple recurrent neural network (RNN) models trained on the same tasks develop similar representations (Botvinick and Plaut, 2006; M. Wang, Fusi, and K. Stachenfeld, 2025; J. C. Whittington, William Dorrell, et al., 2025). This prompts a normative question, why do both PFC and RNNs learn this representation? And what determines the intricate structure of these representations, for example, why are the subspaces are aligned in some tasks (Y. Xie et al., 2022), but orthogonal in others (Panichello and Buschman, 2021)? Or why do the sizes of the subspaces differ? Here we develop a normative theory to reason about these schematic memory representations. Our theory studies the optimal representation for structured working memory tasks, under biologically relevant constraints. By studying the optimal codes applied to different settings, we are able to explain representational differences via key choices in task structure. For example, subspace-alignment arises from the correlations between memories; while the sizes of different encoding subspaces can be explained via their proximity to recall. Similarly, we are able to reason about the single neuron representation of tasks, such as why some representations include disjoint populations of neurons encoding different task variables while others only include mixed-selective populations (Basu et al., 2021; El-Gaby et al., 2024). In sum, we are able to precisely frame the computation that might arise from these representations, and use it to reason about how neurons should instantiate such computations. Further, the theory is structurally identical to recent normative grid cell theories (Will Dorrell et al., 2023), demonstrating its use as a shared normative framework for reasoning about the cortical implementation of algorithms and representations. We hope it will serve as a useful theoretical tool as the field uncovers more about these representations.

10.1 Introduction

Many of the brain's higher level functions are ascribed to the prefrontal cortex. In this work we focus on two of them: working memory and schemas. On the one hand, the evidence linking prefrontal cortex with working memory is strong: not only do prefrontal lesions selectively disrupt working memory (Curtis and D'esposito, 2004; D'Esposito and Postle, 2015), but prefrontal neural activity during delays in memory-dependent tasks encodes the remembered items(Funahashi, Chafee, and Goldman-Rakic, 1993; Fuster and Alexander, 1971; Kubota and Niki, 1971). On the other hand, prefrontal cortex is thought to represent the world in a structural

or schematic way, organising information according to the patterns of behaviour in a given situation (Baldassano, Hasson, and Norman, 2018; Bein and Niv, 2025). Train journeys share a behavioural schema—acquire ticket, find platform, find seat, etc.—that differs from restaurants, and the prefrontal cortex encodes the world in a way that reflects these underlying structures. Similarly, not only is there lesion evidence for PFC's role in structured behaviour (Penfield and Evans, 1935), but neural data implicates PFC in encoding the structure of tasks though often in confusing ways (Constantinescu, O'Reilly, and Behrens, 2016; Samborska et al., 2022; Schuck et al., 2016; Walton et al., 2010; J. Zhou, M. P. Gardner, et al., 2019; J. Zhou, Jia, et al., 2021).

Building on decades of neural measurements (Mushiake et al., 2006; Ohbayashi, Ohki, and Miyashita, 2003; Shima, Isoda, et al., 2007), recent work has started to build an understanding of the neural representations of tasks with both working memory and schematic components. Y. Xie et al. (2022) present an illustrative example. In their task, monkeys observe a sequence of dots on a screen, sampled from the corners of a hexagon, and, after a delay period, are tasked with recalling the sequence (via saccades) in the order it was presented (or in reverse order in subsequence versions (Tian et al., 2024)). This task cleanly embodies both functions we're described: the presented dots must be remembered (working memory) in the right order (the task schema). During the delay period, medial prefrontal neural activity decomposes into subspaces - each subspace encodes the memory of a particular sequence element. Critically, the subspaces are structured: each encodes the memory of the first, second, or third dot location—for example there will be one subspace for 'item presented second'. In this case, the task is a simple sequence, so the sequential structure of the subspaces reflect this; but in general the structuring of these subspaces could reflect more complex relations like 'above' (Panichello and Buschman, 2021; Schwartenbeck et al., 2023). These memory subspaces, structured by the underlying task, are a beautiful neural fusing of the two functions—memory and schemas—and are present in a variety of other tasks (El-Gaby et al., 2024; Mushiake et al., 2006; Panichello and Buschman, 2021; Shima and Tanji, 2000), thus suggesting a general neural algorithm for structured working memory.

Supporting this, computational modelling has found that RNNs trained on structured working memory tasks also develop these structured subspace representations (Botvinick and Plaut, 2006; Piwek, Stokes, and Summerfield, 2023; M. Wang, Fusi, and K. Stachenfeld, 2025; J. C. Whittington, William Dorrell, et al., 2025). These results suggest that these subspaces are, in some way, optimal for solving structured memory tasks in recurrently connected neural networks. However, there remain many puzzling representational features, both in the brain and our neural network models. For example, why is the encoding sometimes overtly compositional—with a subspace of activity encoding the first sequence element independent the rest of the sequence (Y. Xie et al., 2022)—while at other times not - with single neurons activate only for a particular sequence, an encoding that depends on all elements of the sequence (Shima and Tanji, 2000)? And even when the encoding is compositional, why do the subspaces sometimes align (Y. Xie et al., 2022), while at other times they are orthogonal (Panichello and Buschman, 2021); why do they differ in size (Y. Xie et al., 2022), but not always (Panichello and Buschman, 2021)?

In this work we develop a normative theory of structured working memory to answer these questions. It takes the form of an efficient computing theory: we frame a set of constraints that the representation has to satisfy to solve the task, then study the biologically optimal task-solving representation. In particular, we introduce a simplified network model, gated linear RNNs, and show that, to solve structured working memory tasks, its representation must construct structured subspaces. We then incorporate biological ideas, mainly nonnegative neural activity and minimising neural firing, and study the optimal efficiently computing representation. We find that the interaction of efficiency with task structure is able to postdict otherwise puzzling representational features, for example, we outline an axis of variation between the most compositional, orthogonal subspace encodings, through a middle setting of aligned subspaces all the way to sequence specific coding. The optimal representation's position on this axis depends on the structure of the task: if memories are completely independent orthogonal subspaces are optimal, whereas if knowing one memory determines another you get sequence specific codes, finally, if one memory is correlated with another than their subspaces align. This explains the aligned subspaces in Y. Xie et al. (2022) and the orthogonal subspaces in Panichello and Buschman (2021) since they sample their sequences without and with replacement respectively, thus creating dependencies (or none) between memories.

Our theory follows an 'efficient computing hypothesis' (ECH): just like classic efficient coding approaches, we are asking the population to encode memories efficiently (the working memory role of PFC), however, in addition the representation must recurrently perform the relevant computation (tracking and recall of structured memories). This theory is mathematically tractable and quite simple, but still able to capture the interesting measured representational structure. Further, it is structurally identical to our recent work on grid cells, suggesting a shared normative framework. As such, we hope it will be useful in thinking about similar neural or algorithmic puzzles, and highlighting the relevant interactions of task and representation.

We proceed as follows. First, we formalise the structured working memory tasks we study, followed by our neural optimisation problem, comprising the constraints of solving the task and our biological efficiency criteria. Then, in section 10.4, we show that, for our model to solve the task, it must use neural subspaces, and provide

some intuition for how their structure. In section 10.5, we then relate the alignment between subspaces to an axis of task-diversity, and show it can roughly explain broad patterns observed in data. Further, we show that our understanding subspace structure is useful beyond prefrontal structured working memory, by using it to explain some modelling results from the motor cortical literature. In section 10.6 we examine the sizes of the encodings of different memories, explaining their structure using the an interaction of delay structure and energy efficiency. In section 10.7 we then consider single neuron structure, asking how we should expect these population structures to manifest in single neuron tuning curves. Finally, in section 10.8 we consider the implementation of qualitatively different 'algorithms', and attempt to reason about their benefits and measurable signatures. We then conclude with links between our proposed model and existing work in this area.

10.2 Task Formalism and Optimisation Problem

We begin by describing the tasks we're studying and a rough description of our normative theory of neural representations.

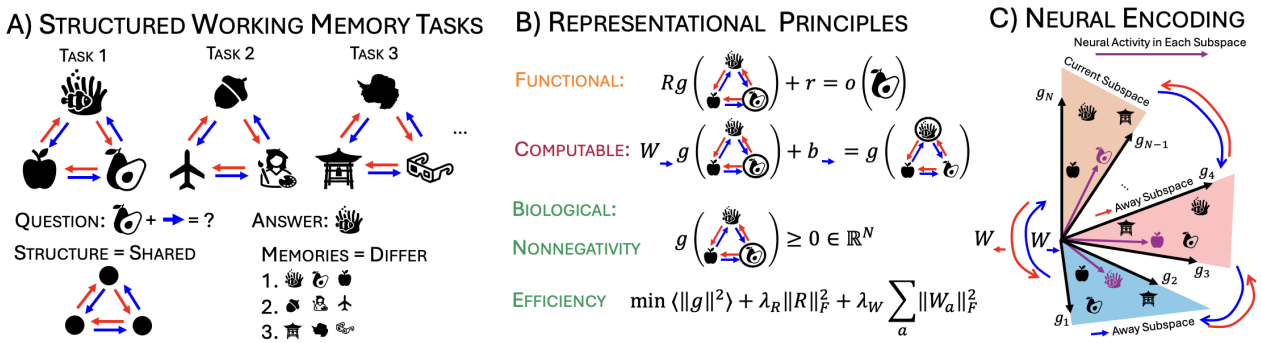


Figure 10.1: **A:** We study the representation of tasks with both working memory and schematic components. The schematic component is an underlying structure shared across tasks - a graph - combined with a set of actions that move you around it. We depict an example task with three states in a loop and two actions, red and blue, that move you in opposite directions around the loop. The working memory component is that, across tasks, the state-stimuli pairing changes. In a new task the representation experiences each state-stimuli pair once; then when it returns to the same state it has to recall the stimuli paired to that state. **B:** We mathematical encode these ideas into an optimisation problem over representations. The representation is a vector of neural firing rates, $\boldsymbol{g}$, that is a function of both the current tasks stimuli-state pairing and where in the graph you are. To solve the task the representation needs to satisfy two constraints. First, a functional constraint, it must be possible to decode the memory at the current position from the representation; we do this with a linear readout. Second, a computability constraint, when you take an action the representation must update to predict the new state's memory; we do this with action-specific matrices. These two specify a representation that can solve the task. Our third constraints selects which task-solving representation to use, we choose the most efficient biological representation by constraining the firing rates of the neurons and minimising energy use by either neural activity or weights. **C:** Representations that solve this task are structured around a series of subspaces as in the data (J. Chen et al., 2024; Panichello and Buschman, 2021; Tian et al., 2024; Y. Xie et al., 2022), each encodes the memory of the stimuli a particular action away. For example, in the depicted task there is a readout subspace encoding the current stimuli, a subspace encoding the stimuli one blue action away, and another one red action away. The position of the neural activity in each subspace encodes the stimuli associated to that state. Taking an action corresponds to shuffling the activity in the subspaces, for example, as depicted, when you take the blue action the activity in each subspace moves following the blue arrows.

Task Formalism. We introduce a general structured working memory formalism that encapsulates existing experimental studies. Sequence memory tasks consists of items that must be recalled (e.g. three items ABC) and a rule (e.g., in reverse) prescribing the ordering they must be recalled in. Each task consists of many different sequences, each sharing the same rule, but with different items to be remembered (i.e., A, B, & C serve as placeholders and one sequence could have $\{A = \text{Apple}, B = \text{Coral}, C = \text{Avocado}\}$ and another $\{A = \text{Peach}, B = \text{Wolf}, C = \text{Planet}\}$). For nomenclature, *task* describes the generic version (ABC) and the underlying rule, whereas *problem* describes a particular sequence instantiated from that task (Apple, Coral, Avocado etc).

The underlying rules are not always as simple as recalling a sequence either in forwards or backwards order. We can construct a more general framework by thinking about sequences as drawn from a walks on a graph.

In this case the rule is the structure of the graph, and different problems are the same graph structure but with different node-stimulus pairings. For example, the graph could be the loop ABC, and the walks could be four repeats of this loop: $ABCABCABCABC$. After seeing the first three observations, the aim is to predict (recall) upcoming observations. To solve these tasks you need a memory to remember past observations, but also knowledge of the graph connectivity because this determines which observation comes next.

In general, graph structures can be more complicated than loops, and the walks don't have to be repeating the same trajectory over and over again (trajectories could differ for different problems). To allow for accurate predictions in these cases the agent requires two pieces of information at each node: an observation $\boldsymbol{o}(\boldsymbol{s}) \in \mathbb{R}^d$ at each (unobserved) graph node, $\boldsymbol{s}$, corresponding to the node-stimulus pairing, and an action $\boldsymbol{a}$ that determines the agent's subsequent movement in the graph from state $\boldsymbol{s}$ to state $\boldsymbol{s} + \boldsymbol{a}$. So, the general sequences are as follows $\{(\boldsymbol{o}_{\boldsymbol{s}_0}, \boldsymbol{a}_0), (\boldsymbol{o}_{\boldsymbol{s}_1}, \boldsymbol{a}_1), (\boldsymbol{o}_{\boldsymbol{s}_2}, \boldsymbol{a}_2), (\boldsymbol{o}_{\boldsymbol{s}_3}, \boldsymbol{a}_3)...$ where $\boldsymbol{s}_{i+1} = \boldsymbol{s}_i + \boldsymbol{a}_i$. Again, the aim it to predict (recall) upcoming observations $\boldsymbol{o}_t$ based on past observations and actions $\{\boldsymbol{o}_{<t}, \boldsymbol{a}_{<t}\}$. We note the underlying structure of the task is unobserved, as all that the animal, or model, observes is a sequence of observations and actions.

Example tasks. We outline two tasks whose contrasting representations will play a key role in this paper. Y. Xie et al. (2022) study exactly the ABC) loop task, but only with one repeated once ($ABCABC$): $\{(\boldsymbol{o}_1, \texttt{Forwards}), (\boldsymbol{o}_2, \texttt{Forwards}), (\boldsymbol{o}_3, \texttt{Forwards}), (\boldsymbol{o}_1, \texttt{Forwards})...\}$. On the other hand, Panichello and Buschman (2021) present two observations AB and then a velocity signal $\boldsymbol{v}$ which determines whether to recall observation A or B: $\{(\boldsymbol{o}_0, \texttt{Forwards}), (\boldsymbol{o}_1, \texttt{Stay Still}$ or $\texttt{Swap}), (\boldsymbol{o}_1$ or $\boldsymbol{o}_0, -)\}$.

Efficient Computing Hypothesis. We formalise an approach we've been terming the 'Efficient Computing Hypothesis' (ECH). In contrast to the efficient coding hypothesis, which states neural firing can be understood as the most efficient encoding of information, ECH argues that neural representations can be understood as the most efficient implementation of an algorithm. Like efficient coding, ECH takes the form of an optimisation problem, and through a combination of constraints and objectives, representations can be predicted.

The ECH optimisation problem is made up of the following constraints and objectives: First, the right memory must be decoded at the right time (i.e., it solves the task; Functional Constraint); Second, this encoding must be easily computed with (i.e., next step predictions are possible using a consistent linear map; Computable Constraint); Third, the neural tuning (at a single neuron and population level) is biologically plausible. These constraints are natural and describe a neural system that solves the sequential task efficiently is biologically plausible manner. Fortunately, these constraints are also relatively simple, producing a clean model to understand of when and why learned neural representations in brains and machines look the way they do. To avoid overwhelming the reader, we will introduce each part of formalism as it comes into play.

10.3 To solve the task Gated Linear RNNs need Structured Subspaces

First, we show that the two constraints which encode solving the task, the functional - decode relevant memory - and computable - allow movement around states to recall other memories - imply that 1) the representation is equivalent to a gated linear RNN and 2) the representation consists of structured subspaces.

How a Computable & Functional Representation Solves the Task. Solving the task means, first, that a neural representation, $\boldsymbol{g} \in \mathbb{R}^N$, is able to recall the correct item, $\boldsymbol{o}(s)$, at the correct state, s. For example, if the task is $\{A = \text{Apple}, B = \text{Coral}, C = \text{Avocado}\}$ and you're in state B, then $\boldsymbol{g}$ must be able to tell you 'Coral'. To do this, the representation must be a function of both the state-stimuli pairing from the current trial, $\boldsymbol{o}(\boldsymbol{s})$, and which state you're in, denotes $\boldsymbol{g}(\boldsymbol{s}, \boldsymbol{o}(\boldsymbol{s}))$. For simplicity we link the representation $\boldsymbol{g}$ to the prediction $\boldsymbol{o}$ by a readout matrix $\boldsymbol{R}$ and bias $\boldsymbol{r}$, producing the 'functional' constraint:

$$\text{FUNCTIONAL/MEMORY:} \qquad \boldsymbol{R}\boldsymbol{g}(\boldsymbol{s}, \boldsymbol{o}(\boldsymbol{s})) + \boldsymbol{r} = \boldsymbol{o}(\boldsymbol{s}) \tag{10.1}$$

i.e. in state s, the readout matrix has to extract the encoding of the stimuli, $\boldsymbol{o}(s)$, paired to the current state, s.

While this ensures the representation can be used to predict the current state's stimulus, we have not yet linked representations from neighbouring states together. We are currently missing the 'computing' part that allows next step prediction. The ECH encodes this via an 'computable' constraint. This states that the representation at any given state, s, must be able to construct all other representations at other states, s' using the action that gets you from s to s'. This can only be done if the representation is a function of the current state and all state-stimuli pairings: $\boldsymbol{g}(\boldsymbol{s}, \{\boldsymbol{o}(\boldsymbol{s})\})$. Precisely, we require this predictability to be implemented via action dependent matrices, $\boldsymbol{W}_{\boldsymbol{a}}$, corresponding to the actions, $\boldsymbol{a}$, in the sequence.

$$\text{COMPUTABLE/STRUCTURAL:} \quad \boldsymbol{W}_{\boldsymbol{a}}\boldsymbol{g}(\boldsymbol{s}, \{\boldsymbol{o}(\boldsymbol{s})\}) + \boldsymbol{b}_{\boldsymbol{a}} = \boldsymbol{g}(\boldsymbol{s} + \boldsymbol{a}, \{\boldsymbol{o}(\boldsymbol{s})\}) \tag{10.2}$$

This constraint links all representations together via the action matrices, $\boldsymbol{W_a}$, allowing the agent to traverse its internal representation to reach its encoding of any other state. Satisfying the above two equations places large constraints on both $\boldsymbol{g}$ and the connectivity between neurons, $\boldsymbol{W}$. In particular, $\boldsymbol{W}$ (and the bias) must obey the abstract structure of the task. Taking action $\boldsymbol{a}$ followed by its inverse $\boldsymbol{a}^{-1}$ takes you back to the same state, s, and it must it do the same for the representation: $\boldsymbol{W_{a^{-1}}W_a g}(\boldsymbol{s}, \{\boldsymbol{o}(\boldsymbol{s})\}) = \boldsymbol{g}(\boldsymbol{s}, \{\boldsymbol{o}(\boldsymbol{s})\})$. Naturally, different underlying graph structures will require different matrices $\boldsymbol{W}$.

This might seem a peculiar way to implement the representational updates; we choose it for a few reasons. First, as we'll see, it is able to capture the key phenomenon while being simple. Second, making the updates (i.e. the matrices) action dependent, rather than state and action, minimises the number of required transformations and ensures they generalise to tasks with the same structure. For example, if you learn that $\boldsymbol{W}_{\text{forward}}$ is the inverse of $\boldsymbol{W}_{\text{backward}}$ in one task, then you can apply that to all new tasks, thanks to the factorisation of action-update from state. Finally, this formalism is effectively a gated linear RNN, as explained next and in chapter 2, a model has been successfully used for motor cortex (Logiaco, Abbott, and Escola, 2021) and attractor circuits (Will Dorrell et al., 2023; Gao, J. Xie, Zhu, et al., 2019; Issa and K. Zhang, 2012). Gated linear networks have been shown to be good models of ReLU neural networks in other settings (Jarvis, Klein, Rosman, and A. Saxe, 2023; Jarvis, Klein, Rosman, and A. M. Saxe, 2025; A. Saxe, Sodhani, and Lewallen, 2022), suggesting a similar correspondence between our model and standard ReLU RNNs.

Actionability is equivalent to a gated linear RNN with connectivity defined by the task structure. To see this link to gated linear RNNs more clearly, imagine an RNN that has different recurrent weights, $\boldsymbol{W_{a_t}}$, for each action. At each timestep it sees the observation, $\boldsymbol{o}_t$ - if it has already seen this observation it ignores it and carries on, else the observation is new and needs to be stored. We therefore introduce a gate that determines whether this state has not been visited before, which tells the representation whether to ignore this observation or store it via a read-in matrix $\boldsymbol{W}_0$. Mathematically:

$$\boldsymbol{g}_t = \boldsymbol{W}_{\boldsymbol{a}_{t-1}}\boldsymbol{g}_{t-1} + \boldsymbol{b}_{\boldsymbol{a}_{t-1}} + \boldsymbol{W}_0\boldsymbol{o}_t\mathbf{1}_{\{\boldsymbol{s}_t \text{ unvisited}\}} \tag{10.3}$$

Where $\mathbf{1}_{\boldsymbol{s}_t \text{ unvisited}}$ is 1 if $\boldsymbol{s}_t$ has not been previously visited, and 0 otherwise. This means the RNN stores each distinct observation once. In order to do this the RNN has to have at least (number of states) $\times$ (dimensionality of observations) neurons. The RNN makes next time-step predictions via first taking an imagined step into the future and then reading out via a readout matrix $\boldsymbol{R}$:

$$\hat{\boldsymbol{o}}_{t+1} = \boldsymbol{R}(\boldsymbol{W}_{\boldsymbol{a}_{t+1}}\boldsymbol{g}_t + +\boldsymbol{b}_{\boldsymbol{a}_{t-1}}) + \boldsymbol{r} \tag{10.4}$$

If the RNN has solved the task then, similarly to the argument above, $\boldsymbol{g}_t$ must be a function of the current state, s_t, and all state-stimuli parings; $\boldsymbol{g}(\boldsymbol{s}_t, \{\boldsymbol{o}(\boldsymbol{s})\})$. Since this gated linear RNN uses action dependent matrices, it's representations are constrained by the computable constraints we framed earlier (Equation 10.2). Thus our ECH problem just describes the same sent of optimal representations that the above RNN would learn, but it automatically makes the activity as a function of state, s, as opposed to time, t.

In sum, a functional and computable encoding of a structured working memory task is equivalent to the representation in a gated-linear RNN trained on the same task. The benefit of posing the problem as constraints is that it lends itself to mathematical analysis, which we will use to generate our neural predictions.

The representation must have structured subspaces. We have outlined how the connectivity, $\boldsymbol{W_a}$, mirrors the underlying task. We can use this to show that, together, the functional and computable constraints imply structured subspaces. Structured subspaces are a set of subspaces, connected to one another mirroring the underlying structure of the task, that can each simultaneously store a memory. The connectivity structure allows memory representations to be passed between subspaces so that the right memory is in the right subspace to be recalled.

You can see this structure by considering the neural subspace spanned by the read-out weights, $\boldsymbol{R}$. Two things must be true about activity in this subspace, first it must contain the current observation to permit recall, and second it can't contain encodings of the other memories as they would corrupt recall. The other memories must be stored elsewhere in the representation, orthogonal (hidden) to the readout weights $\boldsymbol{R}$, fig. 10.2. Moreover, this 'hidden' neural activity orthogonal to the readout weights has further structure. When taking action $\boldsymbol{a}$, a corresponding weight matrix $\boldsymbol{W_a}$ updates the representation: $\boldsymbol{g}(\boldsymbol{s}_t + \boldsymbol{a}, \{\boldsymbol{o}(\boldsymbol{s})\}) = \boldsymbol{W_a g}(\boldsymbol{s}_t, \{\boldsymbol{o}(\boldsymbol{s})\})$ which must cause the representation's encoding of $\boldsymbol{o}(\boldsymbol{s}_t + \boldsymbol{a})$ to be mapped from somewhere in the hidden space to the read-out subspace so that $\boldsymbol{Rg}(\boldsymbol{s}_t + \boldsymbol{a}, \{\boldsymbol{o}(\boldsymbol{s})\}) = \boldsymbol{o}(\boldsymbol{s}_t + \boldsymbol{a})$ (We're ignoring biases for the moment, see APP for full treatment). Since the transformation is linear, there must exist a subspace in the hidden space where memory of the observation $\boldsymbol{a}$ away from your current location is stored. Following the same logic, there must exist different subspaces for all different $\boldsymbol{a}$ (and indeed sequence of actions $\boldsymbol{a}$) from your current location, with the contents of each subspace answering the hypothetical 'what would I observe if I took action $\boldsymbol{a}$ (or

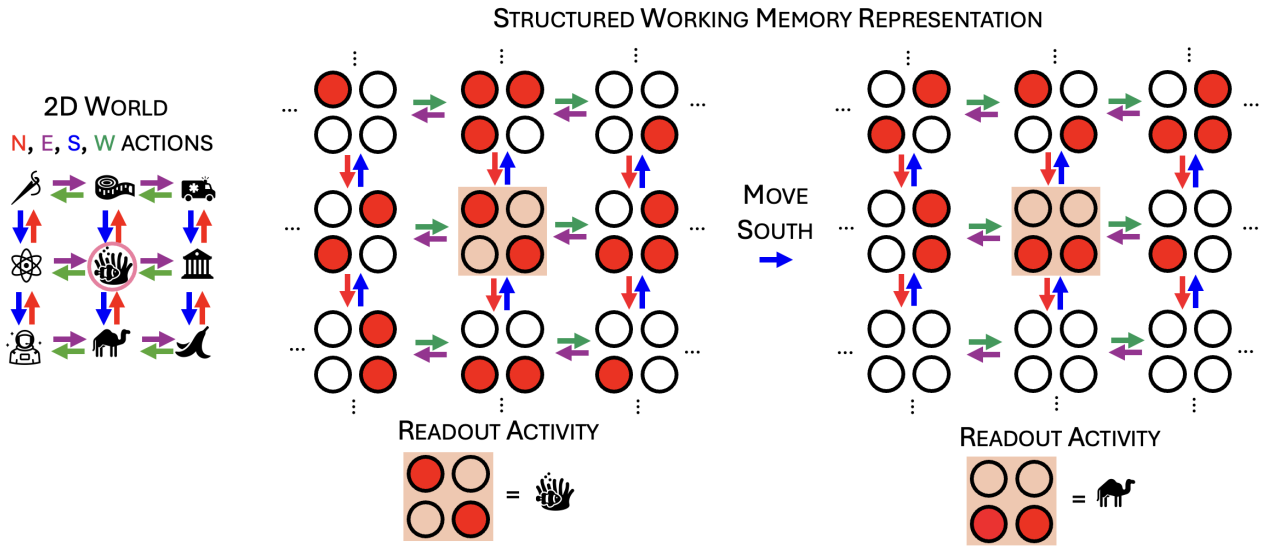


Figure 10.2: We give some intuition for the subspace structure of the representation. **Left:** Imagine we are in a 2D world with actions N, S, E, W. **Middle:** Circles represent neurons that are either active (filled) or inactive. The readout subspace comprises the four highlighted neurons. Activity in this subspace drives the prediction of 'Coral'. **Right:** But, when taking an action, e.g. 'south', activity encoding the memory one step south, the camel, is shuffled into the readout subspace from its previous hiding place, the '1 south' subspace. Similarly, subspaces exist for all other actions, and sequences of actions, storing the memory according to the offset from the current position. When an action is taken all these memories shuffle according to the rules of the task (notice the swapping of the direction of the arrows).

sequence of actions $\boldsymbol{a}$)' (we label each subspace as $U^{\boldsymbol{a}}$). These subspaces must be connected to one-another by recurrent weights, $\boldsymbol{W}_{\boldsymbol{a}}$, according to the structure of the underlying task. For example, taking action $\boldsymbol{a}'$, moves the contents of all subspaces $U^{\boldsymbol{a}}$ to their 'neighbouring' subspace $U^{\boldsymbol{a}+\boldsymbol{a}'}$, fig. 10.2. With these subspaces, we can summarise the underlying representation as

$$\boldsymbol{g}(\boldsymbol{s}, \{\boldsymbol{o}(\boldsymbol{s})\}) = \sum_{a} U^{\boldsymbol{a}} \boldsymbol{o}(\boldsymbol{s} - \boldsymbol{a}) \tag{10.5}$$

As such, we define each subspace using the matrix $U^{\boldsymbol{a}} \in \mathbb{R}^{N \times d}$ which maps the observation at action, $\boldsymbol{a}$, away into a specific subspace of neural activity. The activity in each of these subspaces is then shuffled around according to the rules: i.e. $U^{\boldsymbol{a}+\boldsymbol{a}'} = \boldsymbol{W}_{\boldsymbol{a}'} U^{\boldsymbol{a}}$. To see that this representation, $\boldsymbol{g}$, has an associated algorithm in which memories are passed between subspaces consider how taking the action $\boldsymbol{a}'$ updates the representation:

$$\boldsymbol{g}(\boldsymbol{s} + \boldsymbol{a}', \{\boldsymbol{o}(\boldsymbol{s})\}) = \boldsymbol{W}_{\boldsymbol{a}'} \sum_{a} U^{\boldsymbol{a}} \boldsymbol{o}(\boldsymbol{s} - \boldsymbol{a}) = \sum_{a} U^{\boldsymbol{a}+\boldsymbol{a}'} \boldsymbol{o}(\boldsymbol{s} - \boldsymbol{a}) = \sum_{a} U^{\boldsymbol{a}} \boldsymbol{o}(\boldsymbol{s} - \boldsymbol{a} - \boldsymbol{a}') \tag{10.6}$$

So, after the action of $\boldsymbol{W}_a$ each subspace contains a different memory. Lastly, we note that, from Equation 10.1, the readin and readout subspace must be the subspace corresponding to $\boldsymbol{a} = 0$. If the subspaces are linearly independent, this representation can solve all problems within a given task (i.e. any sequence of observation and actions); arbitrary observations can be read-in in to the readin/out subspace thus any sequence of observations can be handled, and any sequence of actions can be used to pass memories between subspaces, all the while ensuring the right memory is in the right subspace at any point in time. As such, this subspace passing algorithm matches the 'slots' (different word, same thing) observed in nonlinear recurrent neural network models trianed on similar tasks (Botvinick and Plaut, 2006; Piwek, Stokes, and Summerfield, 2023; M. Wang, Fusi, and K. Stachenfeld, 2025; J. C. Whittington, William Dorrell, et al., 2025) and similarly matches the working memory subspaces observed in prefrontal cortex in a variety of tasks (El-Gaby et al., 2024; Panichello and Buschman, 2021; Y. Xie et al., 2022). This is despite the fact that standard nonlinear RNNs (and presumably the brain) are much more expressive, and could have learnt all sorts of different algorithms.

Before embarking on using this subspace framework to make neural predictions, we note one caveat. In our work the notion of action is performing a lot of work. Thus far we have interpreted the actions as movements around a graph of states, and the corresponding action matrices as shuffling the subspaces appropriately, fig. 10.2. However, there might be multiple different ways to decompose the same task into actions. For example, in our framework the easiest way to formalise the task of Y. Xie et al. (2022) - recalling a sequence of three presented dots - is using a single action that loops you periodically through the three memories. In reality, the task has

delay periods, sequences of different length, and later the animals were trained to reverse these sequences (Tian et al., 2024). As such, a more reasonable choice would instead be to frame the actions as concepts like 'first cue presentation', 'delay', 'go cue', 'second memory recall', etc. We explicitly consider these parametersiations in section 10.8. Different choices, since they embody a different structure, lead to different subspace shuffling algorithms, and make slightly different neural predictions, allowing us to infer which algorithm is at work in the measured neural data from Y. Xie et al. (2022). However, in all parameterisations, the same logic that we are about to present regarding the structure of the subspaces remains true. For now, we therefore proceed with the theoretically simplest case of movement around a graph, which we use to illustrate the core findings and match neural data.

10.4 Coupling of task statistics and efficiency determine subspace structure

Eq. 10.5 tells us the solution that (gated linear) RNNs learns in the structured memory problems. However, this solution can be in any basis (i.e., it is valid up to an invertible linear transformation: $\boldsymbol{W}_a \to \boldsymbol{T}\boldsymbol{U}\boldsymbol{T}^{-1}$, $\boldsymbol{R} \to \boldsymbol{R}\boldsymbol{T}^{-1}$, $\boldsymbol{W}_o \to \boldsymbol{T}\boldsymbol{W}_o$, where $\boldsymbol{T}$ is an invertible linear transformation). This means we know nothing, yet, about how these structured subspaces are actually implemented in neurons. Following efficient coding arguments, we will think about how this computation can be implemented most efficiently implemented given the statistics of the tasks under consideration.

Therefore, we now aim to understand how task statistics conspire to structure the optimal neural representation, i.e., how the space of invertible linear transformations gets constrained by task statistics such that particular, and often meaningful, bases are preferred. To do so we must introduce the final set of constraints in the ECH; the biological constraints of nonnegativity and energy efficiency:

$$\text{BIOLOGICAL NONNEGATIVITY:} \quad \boldsymbol{g}(s, \{\boldsymbol{o}(s)\}) \geq 0 \tag{10.7}$$

$$\text{BIOLOGICAL EFFICIENCY:} \quad \mathcal{L} = \underbrace{\langle ||\boldsymbol{g}(s, \{\boldsymbol{o}(s)\} - \theta||_2^2 \rangle_{s, \{\boldsymbol{o}(s)\}}}_{\text{Firing Energy}} + \underbrace{\lambda_R ||\boldsymbol{R}||_F^2 + \lambda_W \sum_a ||\boldsymbol{W}_a||_F^2)}_{\text{Weight Energy}} \tag{10.8}$$

Where θ represents a target firing rate from which the neural activity is penalised for deviating (following evidence that neurons might aim for a firing rate set-point (Chintaluri and Vogels, 2023)). We see that the first three constraints (Equations 10.1, 10.2, & 10.7) are exact constraints, while the last (Equation 10.8) is soft. Thus, in order to discover the optimal representations in ECH, we minimise the soft constraint (10.8) subject to the exact constraints (Equations 10.1, 10.2, & 10.7).

We already know that the first two constraints imply structured subspaces. The remaining constraints will just shape the exact neural implementation. In general it is hard to solve the full problem analytically and our only successes have been with simple ansatz in a small number of cases. However, dropping the nonnegativity constraint simplifies the problem significantly, providing intuition that matches much behaviour of the full system. This intuition can be seen by considering the subspace dot-product matrix, constructed by stacking subspace matrices:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_0 & \boldsymbol{U}_{a_1} & \dots & \boldsymbol{U}_{a_{A_1}} \end{bmatrix} \in \mathbb{R}^{N \times AN_O} \qquad \boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{U} \in \mathbb{R}^{AN_O \times AN_O} \tag{10.9}$$

where A denotes the number of subspaces and N_O the dimensionality of the output space (and N is the dimensionality of the neural representation). A brief side-point, the meaning of the functional and computable constraints are particularly clear in this parametrisation: if the matrix $\boldsymbol{U}$ has linearly independent columns then the representation can be used to solve the task. Now, the dot-product structure, $\boldsymbol{D}$—where each block $\boldsymbol{D}_{ij} \in \mathbb{R}^{N_O \times N_O}$ describes the dot product (similarity) between the i^{th} and j^{th} subspaces—plays a key role in determining the optimal representation. For example, having dropped nonnegativity we can write the entire loss as follows (and with nonnegativity life is not so different):

$$\mathcal{L} = \underbrace{\text{Tr}[\boldsymbol{D}\Sigma_O]}_{\text{Activity Energy}} + \underbrace{\lambda_R \,\text{Tr}[\boldsymbol{D}_{00}^{-1}]}_{\text{Readout Weight Energy}} + \underbrace{\lambda_W \sum_{a \in N_A} \text{Tr}[\boldsymbol{P}_a^T \boldsymbol{D} \boldsymbol{P}_a \boldsymbol{D}^{-1}]}_{\text{Recurrent Weight Energy}} \tag{10.10}$$

where we've introduced two types of task-specific constant matrices, Σ_O and $\boldsymbol{P}_a$. The first, Σ_O, simply measures the correlation across all sampled problems between the memories in each subspace. The second, the $\boldsymbol{P}_a$ matrices, implement the relevant shuffling operation, for example, $\boldsymbol{P}_{\text{forward}}$ takes the activity from one subspace, a, and

moves it to the next (the equivalent of what the action matrices do). In maths, using A different N_O dimensional blocks:

$$\boldsymbol{P}_{\text{forward}} = \begin{bmatrix} \mathbf{0} & \mathbb{1} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{1} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} \\ \vdots & & & \ddots & \vdots \\ \mathbb{1} & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} \end{bmatrix} \tag{10.11}$$

Much of the structure of the problem by studying the fixed point condition in various limits:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}} = \Sigma_O - \lambda_R \boldsymbol{D}^{-1} \mathbb{1}_{00} \boldsymbol{D}^{-1} + \lambda_W \sum_a (\boldsymbol{P}_a^T \boldsymbol{D}^{-1} \boldsymbol{P}_a - \boldsymbol{D}^{-1} \boldsymbol{P}_a^T \boldsymbol{D} \boldsymbol{P}_a \boldsymbol{D}^{-1}) = 0 \tag{10.12}$$

Recurrent Weights Prefer Orthogonal and Identically-Sized Subspaces. First, consider the limit λ_W very large, allowing us to ignore the first two terms. We find that the following are solutions:

$$\boldsymbol{D} \approx \alpha \mathbb{1} \tag{10.13}$$

for a positive constant $\alpha > 0$. This tells us that the recurrent weight loss prefers all subspaces to be the same size and orthogonal. This can be intuitively understood: first, if subspaces differ in sizes than the weight matrices have to 'do work' growing and shrinking the activity as they move around, fig. 10.3. Shrinking a memory as it moves from readout to a 'hidden' subspace is cheap, but the recurrent weights also have to do the reverse, growing the 'hidden' memory as it moves to the readout. The growing operation is always more costly than the shrinking is cheap, resulting in a cost for changing subspace sizes. Second, if the subspaces are aligned the weights have to be larger to tease apart the two, fig. 10.3.

Firing Efficiency Aligns Statistically Correlated Memories, Whitening the Representation. Now consider the regime when λ_W is very small, allowing us to drop the last bracket. Then the solution is:

$$\boldsymbol{D} \approx \sqrt{\lambda_R} \mathbb{1}_{00} \Sigma_O^{-\frac{1}{2}} \tag{10.14}$$

This is telling us that all the subspaces are basically empty except the readout subspace ($\boldsymbol{D}$ is zero except in the first block). The size of this subspace is set by the size of the targets ($\Sigma_O^{-\frac{1}{2}}$) and the ratio of readout to activity costs (λ_R). The higher the readout weight costs, the larger the subspace needs to be (bigger $\boldsymbol{D}$) to allow the readout weights to correspondingly shrink. All the other non-readout subspaces are negligibly small. This incurs a large recurrent weight cost, as in the previous section, but this doesn't matter, since λ_W is very small.

Internally, the subspace is structured by the correlational structure of the memories, $\Sigma_O^{-\frac{1}{2}}$. To get some intuition, momentarily forget about sequence memory and consider a neural encoding of two variables, x and y, from a dataset $\mathcal{D}$ of many x and y pairs:

$$\boldsymbol{g} = x\boldsymbol{u} + y\boldsymbol{v} \tag{10.15}$$

Here x is encoded along $\boldsymbol{u}$, and y along $\boldsymbol{v}$ ($\boldsymbol{u}$ and $\boldsymbol{v}$ are two subspaces!). When considering the firing energy of this population it becomes clear that correlated variables (memories) change how the subspaces are aligned:

$$\langle ||\boldsymbol{g}_t||^2 \rangle_t = \langle x_t^2 \rangle_t ||\boldsymbol{u}||^2 + \langle y_t^2 \rangle_t ||\boldsymbol{v}||^2 + 2\langle x_t y_t \rangle_t \boldsymbol{u}^T \boldsymbol{v} \tag{10.16}$$

We see that the final term in above loss depends on the correlation of the two variables (across all tasks) and can be reduced by by anti-aligning their encodings if the variables are positively correlated, and vice versa, fig. 10.3. Thus, minimising activity energy can lead to aligned subspaces when variables are correlated. This is a whitening operation, similar to those derive by standard efficient coding arguments (Atick and Redlich, 1990, 1992).

Finally, we can see an interpretable effect of cross-subspace correlations. If the different memories are statistically independent Σ_O is block diagonal. Studying the fixed point condition, eq. (10.12), we notice that inserting a block diagonal $\boldsymbol{D}$ - a representation orthogonal subspaces - solves all the off-block-diagonal parts of the fixed point condition, leaving a simple condition on the diagonal blocks:

$$\boldsymbol{D}_{kk} (\Sigma_{O,kk} + \lambda_W \sum_a \boldsymbol{D}^{-1}_{f_a(k), f_a(k)}) \boldsymbol{D}_{kk} = \lambda_R \delta_{k0} \mathbb{1} + \lambda_W \sum_a \boldsymbol{D}_{f_a(k), f_a(k)} \qquad \forall k \tag{10.17}$$

where the function $f_a(k)$ is the subspace that subspace k is mapped under action a. As long as these nonlinear matrix equations are solvable (which we conjecture is the case), uncorrelated memories are stored in orthogonal subspaces.

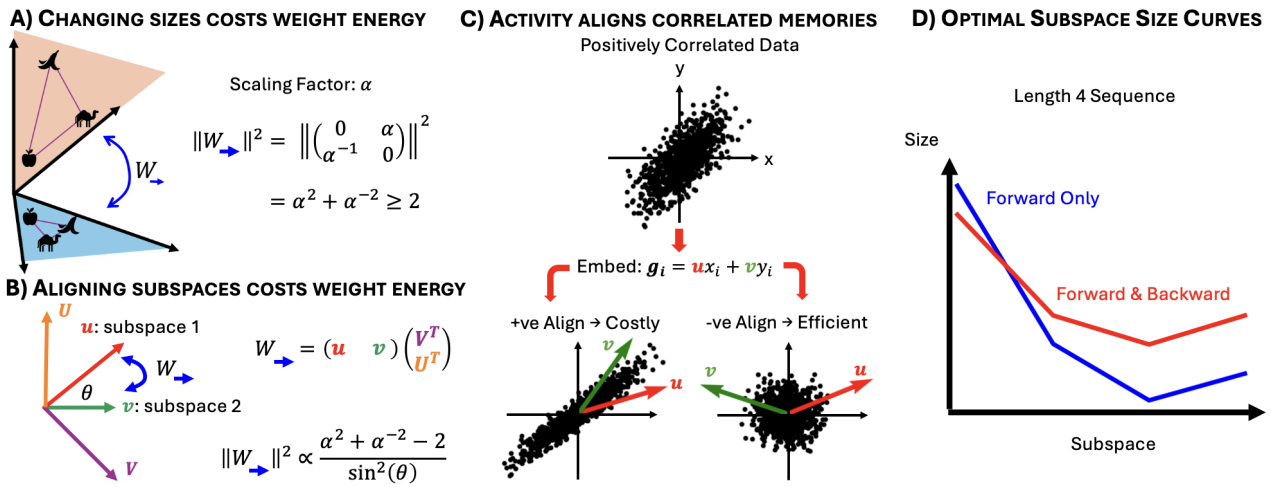


Figure 10.3: The recurrent weight energy is complex, but behaves in two interpretable ways. **A:** First, growing and shrinking activity as it moves between subspaces incurs a weight cost. **B:** Second, disentangling aligned subspaces usually incurs an energy cost, since the weight matrix has to be larger to tease apart the aligned patterns of activity. See the $\boldsymbol{U}$ and $\boldsymbol{V}$ vectors that are constructed to do this teasing - $\boldsymbol{U}$ ($\boldsymbol{V}$) pulls out the $\boldsymbol{u}$ ($\boldsymbol{v}$) subspace ignoring the other. The more aligned the subspaces, the larger these vectors are, the higher the costs tend to be. The logic is more complex than this (you can see from the equation that if the subspaces have the same size, the cost is independent of angle, but this effect doesn't generalise to higher-dimensions), but generally aligning is costly for the recurrent weights. **C:** Aligning the encodings of positively correlated variables is costly, since more of the activity is far from the origin, while negatively aligning saves energy. **D:** These effects compete, producing graceful subspace size decay curves - the readout is largest, other subspaces are smaller in a smooth pattern. Further, the pattern depends on the actions in the task, if both forward and backward are present in this length 4 sequence task the curve is symmetric around the readout space, and size decays as a function of distance from readout.

Competing Costs Structure the Representation, Setting Subspace Sizes. These different effects compete: the activity term shrinks all the subspaces, structures their internals (the activity within each subspace) to whiten the stimulus distribution, and tries to align correlated memories. The readout loss will grow the size of the readout subspace and orthogonalise it from the others. The recurrent weight loss will orthogonalise every subspace, and push their sizes to be the same. These losses fight, both in the tussle to set subspace alignment (activity aligns correlated memories, readout and recurrent weights orthogonalise subspaces), but also in settling the eventual size of the subspaces, as we explore in more detail now.

Intuitively, we will see a few effects. First, the readout space is always largest, since it is the one closest to producing output. Subspaces are then shrunk depending on how 'far' they are from the readout. If a subspace is mapped by one action to the readout then it should be smaller, to save activity energy, but not too small, to save the weight energy of growing and shrinking. But a subspace that only reaches the readout after two actions can afford to be smaller, since the growing-shrinking weight cost can be split over two steps.

To see this mathematically we consider the case where $\Sigma_0 \propto \mathbb{1}$, i.e. each memory is independent. Inserting this condition into eq. (10.12), we find the optimal $\boldsymbol{D}$ contains scaled identity subblocks: each subspace is white but differently sized. These sizes satisfy fixed point equations determined by the action matrices; for example, if there is one action matrix that loops forward through a sequence of length K, the sizes of each subspace σ_k, obey the following fixed point conditions:

$$\boldsymbol{D}^* = \begin{bmatrix} \sigma_0 \mathbb{1} & \mathbf{0} & \dots \\ \mathbf{0} & \sigma_1 \mathbb{1} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \qquad 1 - \delta_{k0} \frac{\lambda_R}{\sigma_k^2} + \lambda_W \left(\frac{1}{\sigma_{k-1}} - \frac{\sigma_{k-1}}{\sigma_k^2}\right) = 0 \quad \forall k = 0, ..., K-1 \tag{10.18}$$

Numerically optimising these parameters produces a graceful decay curve: the readout subspace is largest, and each subsequent subspace gets smaller down to a trough, before then rising again, fig. 10.3. Importantly, the optimal solution depends on the particular task, for example if the problem includes both a forward and backward matrix the size decay curve changes, becoming symmetric.

The effects of these interacting losses are sufficient to explain many observed phenomenon. However, before proceeding to neural data, we briefly highlight the complex role the final part of our problem, nonnegativity, can play in producing meaningful single neuron responses.

Nonnegativity Aligns Range Correlated Variables. By expressing the energy cost in terms of $\boldsymbol{D}$, eq. (10.10), we highlight another aspect of the problem - it is rotationally invariant. If we rotate the subspaces arbitrarily using an orthogonal matrix $\boldsymbol{T}$: $\boldsymbol{U}' = \boldsymbol{T}\boldsymbol{U}$, the dot-product structure, and hence the energy of the representation, stay constant:

$$\boldsymbol{D}' = \boldsymbol{U}'^T\boldsymbol{U}' = \boldsymbol{U}^T\boldsymbol{T}^T\boldsymbol{T}\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{D} \tag{10.19}$$

In our problem, the rotational degeneracy is broken by the nonnegativity. This introduces subtle effects that can lead to aligning the encoding of variables on the basis of the ranges of the two variables, as we'll now explore. Consider the same example as above, but with mean-zero variables x and y (just for illustration). To make $\boldsymbol{g}$ nonnegative we need to add a bias. The minimal choice of bias that doesn't increase firing costs unnecessarily is:

$$\boldsymbol{g} = x\boldsymbol{u} + y\boldsymbol{v} - \underbrace{\min_{\mathcal{D}}[x\boldsymbol{u} + y\boldsymbol{v}]}_{Bias} \tag{10.20}$$

And so the activity energy becomes

$$\langle||\boldsymbol{g}_t||^2\rangle_t = \langle x_t^2\rangle_t||\boldsymbol{u}||^2 + \langle y_t^2\rangle_t||\boldsymbol{v}||^2 + 2\langle x_t y_t\rangle_t\boldsymbol{u}^T\boldsymbol{v} + ||\min_{\mathcal{D}}[x\boldsymbol{u} + y\boldsymbol{v}]||^2 \tag{10.21}$$

We can see that the bias always increases the activity energy (which makes sense, since it is shifting everything to higher firing rates). But the nonnegativity has subtle effects. Consider the case where x and y are statistically independent, then, since $\langle x_t y_t\rangle_t = 0$, the activity loss doesn't prefer one alignment over the other, and the weight costs will push all the subspaces to orthogonalise. But, with nonnegativity, not all orthogonal subspaces are created equal. It turns out that, for range independent variables (where knowing about one variable does not affect the range of the other, but it may change the distribution, i.e., not the same as statistical independence), the optimal orthogonal subspaces are modular, encoding each variable (memory) in a different set of neurons.

To see this consider that, for (range) independent variables, $\min_{\mathcal{D}}[x\boldsymbol{u}+y\boldsymbol{v}] = \min_{\mathcal{D}}[x\boldsymbol{u}]+\min_{\mathcal{D}}[y\boldsymbol{v}]$. Therefore, activity loss resulting from the bias is:

$$||\min_{\mathcal{D}}[x\boldsymbol{u} + y\boldsymbol{v}]||^2 = ||\min_{\mathcal{D}}[x\boldsymbol{u}]||^2 + ||\min_{\mathcal{D}}[y\boldsymbol{v}]||^2 + \underbrace{2\min_{\mathcal{D}}[x\boldsymbol{u}]^T\min_{\mathcal{D}}[y\boldsymbol{v}]}_{\text{Cross Term}} \tag{10.22}$$

The term that depends on the alignment is the cross term. Since the variables are mean-zero, every element of the min vectors, $\min_{\mathcal{D}}[y\boldsymbol{v}]$ and $\min_{\mathcal{D}}[x\boldsymbol{u}]$, is at most 0, and it is only zero when the corresponding element of the subspace vectors, $\boldsymbol{u}$ or $\boldsymbol{v}$ are 0. This means that, for range-independent variables, the minimal value of the cross-term is 0, and this minima is achieved when the subspace vectors have non-overlapping 0 elements, in other words, when each subspace is encoded in a different set of neurons. This has has recently been formalised in William Dorrell, K. Hsu, et al. (2025) where they show precise conditions for modularity based on the statistics of the two variables. In sum though, range independent variables modularise (even when there is statistical dependencies), while range dependent variables can mix.

Translating this back to our sequence memory tasks. If the memories are range independent, i.e., (Panichello and Buschman, 2021), then we expect the subspaces to be orthogonalise, while if the memories are range dependent, i.e., (Y. Xie et al., 2022), then we expect the subspaces to align. In the next sections we use the intuition developed about this problem to outline and understand some patterns in neural data.

10.5 Axis of task diversity explains prefrontal subspace alignment

Many studies have recorded prefrontal neurons encoding memories in subspaces, yet the reported angles between subspaces differ. In particular Panichello and Buschman, 2021 report orthogonal subspaces, whereas Y. Xie et al. (2022) report aligned subspaces. More than that, some studies even report single neuron tuning to a subspace (El-Gaby et al., 2024; Mushiake et al., 2006) while others don't (Panichello and Buschman, 2021; Y. Xie et al., 2022). In this section we sketch out how we can view these tasks as lying on a continuum of memory correlation - more correlated memories lead to more aligned subspaces, independent memories lead to orthogonal subspaces, and single neuron tuning (though not always for reasons our theory cannot explain (Panichello and Buschman, 2021) - see discussion). Further, tasks that don't even report correlations can be put on this axis, since very low task diversity can lead to very aligned subspaces that manifest as sequence specific codes (Shima and Tanji, 2000).

Subspace alignment occurs with range dependent memories. An important difference between the task that reported orthogonal (Panichello and Buschman, 2021) vs. aligned (Y. Xie et al., 2022) subspaces was the memory correlations. The first task sampled items independently, producing maximum task diversity, while the second sampled sequence of dots without replacement, leading to a reduced task diversity.

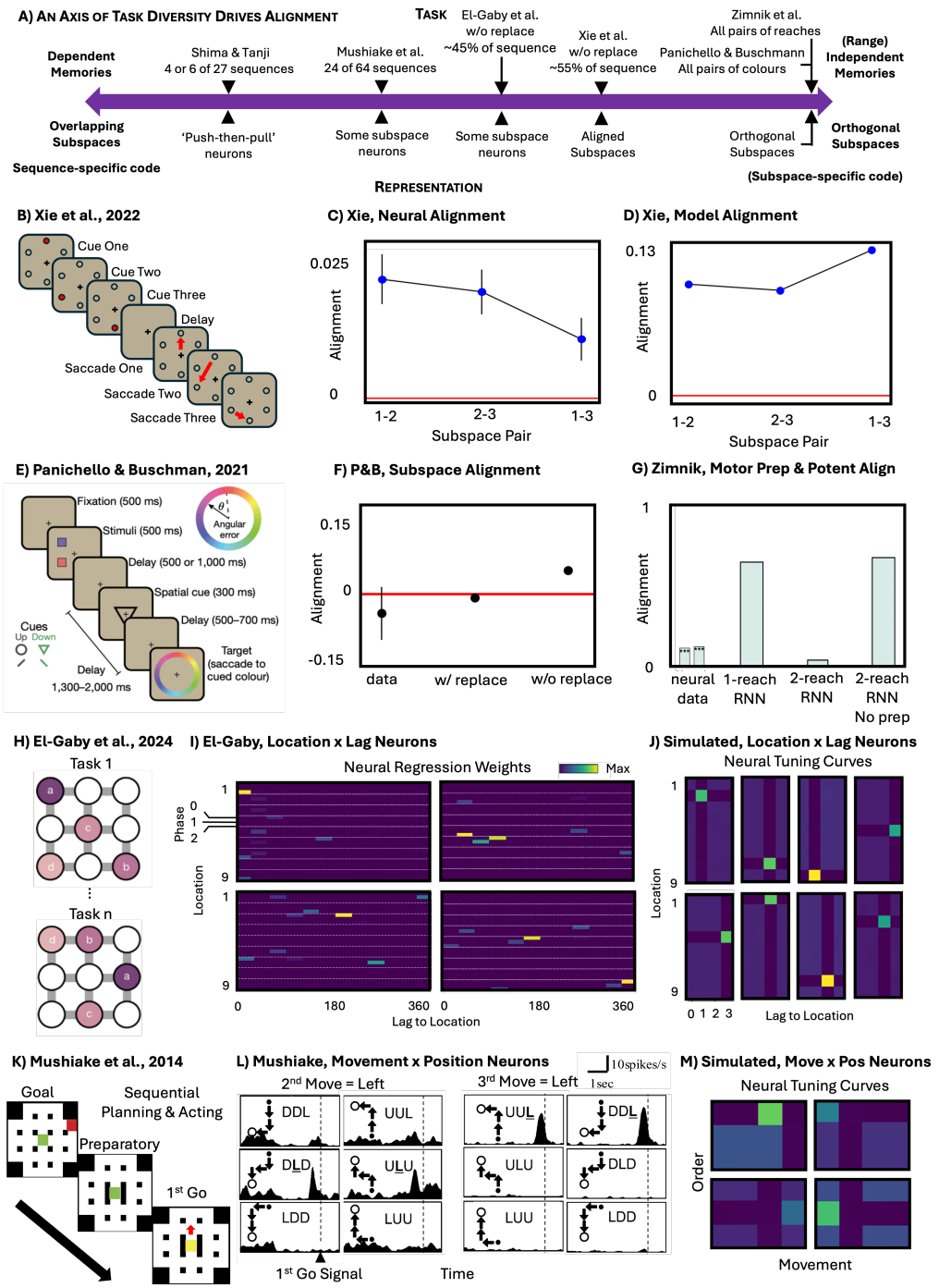


Figure 10.4: **A:** We outline an axis of task and representational alignment. In tasks with independent memories the subspaces are orthogonal, while for correlated memories the subspaces align. In the extreme correlated case the subspaces overlap, producing sequence-specific coding. Various tasks from the literature can be placed on this axis, aligning roughly with their corresponding representation - though our single neuron predictions are not accurate. **B:** The Xie task involves saccading to a remembered sequence of three locations sampled without replacement. This sampling produces correlations that align the encoding subspaces in both **C:** data and **D:** model. **E:** Panichello & Buschman task - two colours are shown one above the other. After a delay, a cue is presented which tells the monkey which colour to saccade to. **F:** Subspaces in both data and model are orthogonal; sampling the colour without replacement produces alignment in the model. **G:** In motor cortex preparatory and potent subspaces are orthogonal. This is observed in RNNs trained to perform a sequence of two delayed reaches, but not a single delayed reach. This can be understood as a reflection of the independence of activity in preparatory and potent subspaces when the preparation of the second reach overlaps with the performance of the first. If, due the task structure, this overlap doesn't occur ('2-reach RNN no prep') the subspaces align again. **G:** El-Gaby et al. train mice to visit a sequence of 4 out of 9 ports and **H:** find neurons tuned to the rewarded location a given number of steps forward in the sequence - i.e. subspace coding neurons. **I:** Our model produces the same effect. **J:** Mushiake et al. train monkeys to perform one of 24 3-move sequences after a delay. **K:** The delay period activity contains neurons tuned to the action a number of steps in the future - subspace coding, as **L:** in our model.

These results are in concordance with optimal representations from the ECH. In order to reduce the firing energy of the representation, independent memories are encoded in orthogonal subspaces (in agreement with Panichello and Buschman (2021)), and range dependent memories are encoded in aligned subspaces (in agreement with Y. Xie et al. (2022)). Indeed we confirm this in simulation: when memories are sufficiently independent we observe orthogonal memories, as in Panichello and Buschman (2021), fig. 10.4F. When we sample memories with correlations, for example without replacement, creating both range and statistical correlations, the subspace align, as in Y. Xie et al. (2022), fig. 10.4D. This makes testable predictions for both tasks; changing sampling with replacement to without replacement, and vice versa, will change the monkey PFC subspace representations from aligned to orthogonal.

More that just orthogonal subspaces, neuron aligned (modular) subspaces are sometimes observed in prefrontal neurons. For example Mushiake et al. (2006) trained monkeys to plan sequences of three actions through a grid world, fig. 10.4K, and observed, during a delay period, individual dlPFC neurons that code for the first action, others the second, and some the third, fig. 10.4L. Similarly, El-Gaby et al. (2024) trained mice to visit a sequence of four locations, fig. 10.4H, and found that some neurons were tuned to the location the mouse would visit next, others to the one after that etc., fig. 10.4I. In our theory range-independent memories are encoding in modular subspaces, producing subspace-tuned neurons. Introducing sufficient correlations leads to alignments and the presence of mixed neurons, the degree of mixed-selectivity roughly correlating with the degree of alignment. As such, relatively range-independent memories as in El-Gaby et al. (2024) and Mushiake et al. (2006), will lead to a propensity for many modular neurons, fig. 10.4J, M. It also interesting that not all neurons reported in El-Gaby et al. (2024) are modular, suggesting it indeed lies somewhere on this axis away from the fully modular end. Modular codes are, however, are not universally observed in places our theory would predict—for example Panichello and Buschman (2021) report mixed-selective neurons even though the memories are independent—and so further work is required to understand single neuron tuning.

Sequence specific coding with low task diversity. As the correlations between memories become extreme the subspaces align further and further. Eventually, the memories are deterministic, knowing the first element uniquely determines the others. In this case the memories are completely correlated, and the subspaces overlap, collapsing to one readout subspace. In this case, rather than a compositional code where each subspace encodes one memory independent of the others, activity in this one readout subspace encodes where and in which sequence the representation currently sits, fig. 10.14.

Some tasks have so few sequences that they lie close to this end, producing single-neuron tuning curves that look more sequence-specific than they do subspace-specific. Indeed, Shima and Tanji (2000) trained monkeys on sequences of three actions, like 'push, turn, pull', but only on either on 4 or 6 of the 27 potential sequences. They found that different neurons encoded 'turn followed by pull' and 'turn followed by push', which is not compatible with a modular subspace model, but does match simulations of very limited diversity tasks with large but not deterministic correlations between memories, fig. 10.14.

Outlook & Similarities to Motor Cortical Phenomenon. While we have not explained all phenomena, we have sketched how a key aspect of task design, the memory diversity, structures the alignment of subspaces and their manifestation in single neurons. We note that this subspace logic seems more broadly useful than PFC structured working memory. A very similar set of findings occur in motor cortex for delayed reach tasks (Kaufman et al., 2016), fig. 10.4G. There preparatory and potent subspaces have been found to be orthogonal but the same subspaces in RNNs trained on similar delayed reach tasks were found to align (Elsayed et al., 2016), fig. 10.4G. Then Zimnik and Churchland (2021) found that training the network to perform two reaches in sequence, one after the other, led it to orthogonalise its preparatory and potent subspaces, why? Since they used standard L2 regularisation techniques, like our efficiency constraint, this can be understood as a correlational effect. If the network only ever performs single-reach tasks the preparatory and potent subspaces are never jointly occupied, the activity in the two subspaces is therefore highly correlated, and they align. Alternatively, dual reach tasks ensure the preparatory subspace must contain an encoding of the second reach while the potent subspace is performing the reach. If the two reaches are independent, the activity in each subspace should be range-independent, and the two subspaces should orthogonalise, fig. 10.4G. As such, motor cortex may orthogonalise the two subspaces since it is regularly performing sequences of unrelated actions. More broadly, it suggests these ideas seem useful wherever neural subspaces appear.

10.6 Subspaces have different activity magnitudes due to energy efficiency

We now explain a second property of prefrontal subspaces—different subspaces encode memories at different magnitudes (more or less neural activity). For example, Panichello and Buschman (2021) find that both

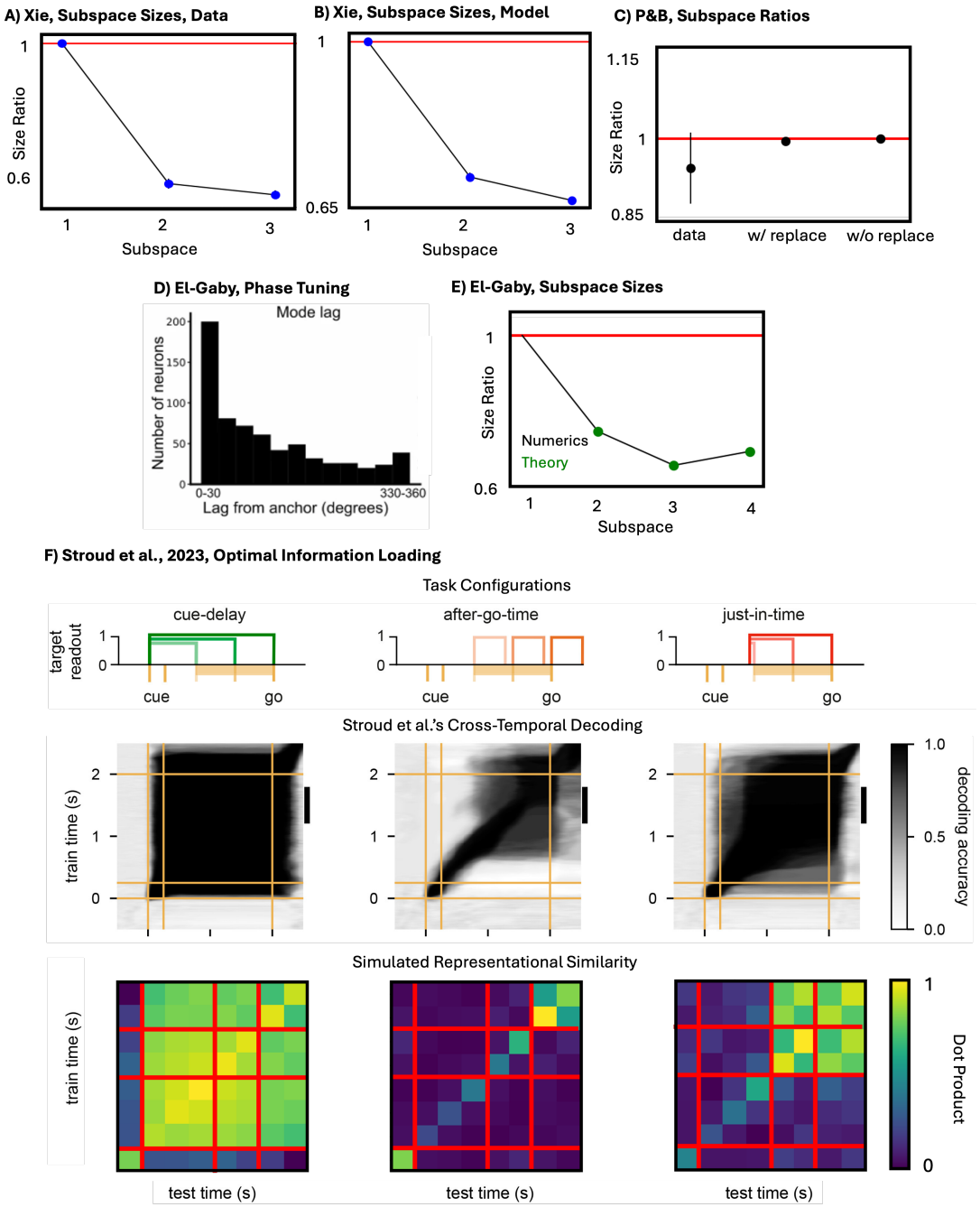


Figure 10.5: **A:** We measure the subspace sizes in Xie et al. and find it decays with distance from the readout, **B:** as in our model. **C:** Contrastingly, the subspace size in Panichello & Buschman are equal, and in our model this is true independent of sampling scheme, since each memory is equally far from the readout. **D:** El-Gaby et al. show a falling then rising distribution of neurons coding for positions a particular distance forward in time. **E:** We don't model continuous 'lag', but even in our four subspace model we see the same effect. **F:** We train a representation to see then recall an observation in different ways. We plot the dot-product similarity of the resulting optimal representation of one memory at different timepoints (bottom row), and compare it to the cross-temporal decoding in RNNs trained by Stroud et al. In the left column the representation must recall the memory from immediately after presentation until the end of the trial. In the middle column the representation has to recall the memory only after a 'go' cue arrives. In the right column the memory must be recalled just prior to the shortest potential delay period until the end of the trial. In the first case (left) the representation is fixed in the readout space. In the second (middle) the representation rotates to keep the size small, before moving to the larger readout space when the go cue arrives. In the third (right) the cue rotates, then at the time of shortest delay, arrives in the readout space in preparation for a potential readout. This leads to a code that rotates for the length of the shortest delay period changing size as it goes, and is then fixed henceforth. Our subspace size logic matches the results of Stroud et al.

subspaces encode the memory with equal activity, while Y. Xie et al. (2022) find that the three subspaces encode memories with different sizes (first memory > second memory > third memory).

Indeed, this is exactly what the ECH predicts for both these tasks, fig. 10.5, due to a balance between weight and activity energy efficiency. In particular for Panichello and Buschman (2021) it says both subspaces should have equal subspaces since they are both one-step away from being readout, where for Y. Xie et al. (2022) the subspaces should gradually reduce in activity the further away from being readout.

We see similar effects in El-Gaby et al. (2024) where subspaces encoding memories further from being 'readout' seemingly have fewer neurons coding for them (a proxy for the size of the subspace). In this task mice run in looping patterns of 4 goal locations, with PFC neurons tuned to the future locations of the mouse. The number of neurons tuned at each offset follows a graceful decline the further in the future the position is, before rising slightly returning to the present, fig. 10.5D. Our simulations show a similar effect, including the slight rise in encoding strength for the most distance memories, fig. 10.5E. This is easily understood, subspaces activity gradually reduces further away from the readout (Lag 0; Subspace 1) to minimise firing energy but as the subspaces get closer again to the readout (e.g., Lag 330; Subspace 4) they increase in activity again gradually to avoid particularly large recurrent weights.

Rotational dynamics during delay periods are energy optimal. Being that subspaces encoding memories 'far' from being recalled have lower neural activity, it is then natural to consider how memories are encoded during delay periods when they are 'far' from being recalled. Our theory predicts that, during a delay period, a single memory will progress though many intermediate subspaces of reducing and then increasing activity as it gets closer to being recalled. Indeed this is exactly what is observed in both our simulations as well as monkey PFC neurons (Stroud et al., 2023), fig. 10.5F.

A critical observation in Stroud et al. (2023) is that when the delay is of fixed (and therefore predictable) duration, the memory rotates through many subspaces until the delay period ends, while with variable delays the memory rotates through the subspaces until until the first time at which the delay period could end and then it stops rotating and remains in the final subspace. This is easily understood; reduce energy during times at which the memory won't be recalled, then leave the memory sitting ready-to-go in the readout subspace even if the delay period hasn't ended (as it might end at any time).

10.7 Single Neuron Tuning and Modularity

Our understanding has so far largely focused on subspace properties— their alignments and sizes—when a neural system optimally solves a single task. In order to build a full understanding of how neural systems implement structured sequence (memory) tasks, there are two further avenues to explore. First we must consider the neural coding of each memory *within* each subspace, and second we must consider what happens when a neural system has to solve more than one structured sequence (memory) task.

In particular we will consider when the neural system is modular or not; when memories have single neuron coding (or not) within each subspace, and when different tasks are solved by different neural populations (or not). In both cases, just like our earlier argument for between subspace modularity, single neuron coding emerges from (range) independence properties in the data distribution.

Modularity within Subspaces. Individual memories can themselves be compositional. For example, Nakajima et al. (2013) train monkeys to perform sequences of two movements where each movement is composed of two parts: an action—pronation or supination—and an arm choice—left or right. Nakajima et al. (2013) found neurons in pre-supplementary motor area that, in line with many mentioned in this work, code for the upcoming actions: some coding for the first action, others for the second. Intriguingly they found these neurons were largely modular *within a subspace*, i.e. neurons mostly coded for either the hand or the action of whichever action they were coding for, fig. 10.6A.

This effect naturally emerges in the ECH if we assume at each subspace needs to encode two memories at the same time; one for 'left' or 'right' and the other 'pronation' or 'supination'. This is the same things at just a single memory being composed out of two parts (and just like the two variables in Equation 10.20. Since all combinations of parts are sampled ('left supination'; 'left pronation'; 'right supination'; 'right pronation'), they are range independent (the variable left/right is range independent from the variable supination/pronation) and therefore, from Equation 10.21, the ECH says that their optimal encoding within a subspace is modular (as well as between subspaces). Indeed we observe this in our simulations, fig. 10.6B. An interesting prediction is that this would disappear were some combinations of the two variables sampled, fig. 10.6C.

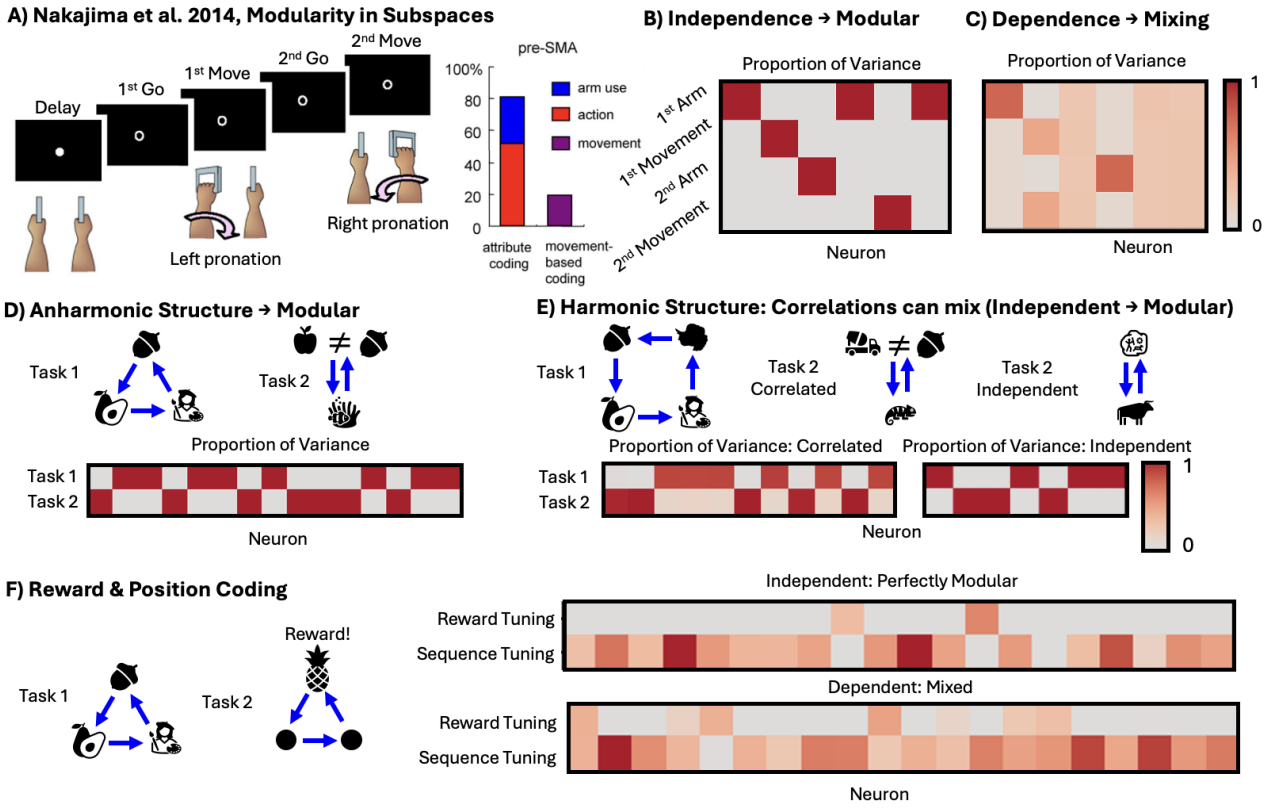


Figure 10.6: **A:** Nakajima et al. trained monkeys to perform a sequence of two movements, each a pronation of supination of one arm. There were neurons coding for both arm movements during the delay period (subspace coding), but interestingly, these neurons tended to code either for arm or action (supination vs. pronation). **B:** This modularity *within* a subspace can be captured by assuming the code is predicting a memory composed of two parts. **C:** However, if these two parts were correlated, their encodings would align, and we would predit there would be fewer of these neurons. **D:** Mixing between tasks depends on both correlations, as explored before, and structural alignment. If the two tasks are not 'harmonically related', which for single loop tasks means they repeat at different frequencies, then the neurons will always be modular, even if the stimuli are correlated. We plot the proportion of each active neuron's explained variance and a function of either all the variables specifying task 1 or 2 (i.e. the set of lagged memories for each task). Each neuron is explained by either task 1 or task 2 variables. **E:** When the tasks are harmonically related (here one task is twice the frequency of the other) correlations can lead the tasks to mix. Independent sequences are still modularly encoded, therefore it is the combination of correlations between the elements of the sequence, and shared structure, that leads to mixing. **F:** We consider two concurrent task 'reward prediction' and 'ABC'. If the two are independent, like position and reward in El-Gaby, the code is modular. If, however, some memories don't occur one away from reward, as in Basu, we get mixed coding, and in fact all 'reward neurons' are also tuned to the a sequence subspace.

Modularity in multiple tasks. Animals do not just solve one task at a time, but many. These problems do not always have the same task structure and so a neural solution to one task may not transfer to another. Even without transfer, the two tasks may be solved by overlapping neural populations (mixed coding), or equally, the two tasks could be coded for in separate neural populations.

To make this more concrete, let us consider a simple setting with two concurrent sequences, in the first memory three observations must be repeated on a loop ($ABCABCABC...$ where A,B,C are different observations for different problems; like Y. Xie et al. (2022)) in the other its only two ($EFEFEF...$). Clearly these two tasks could be solved with two separate populations; one for the ABC task and one for EF. The ECH, however, states the populations may mix if (1) there are correlations in the memories, (2) the sequence structures are 'harmonically related'. This second condition was explored in William Dorrell, K. Hsu, et al. (2025); it means that if the frequency at which one stimulus repeats is not a multiple of the other, then the two tasks will be encoded in different neural populations. Thus, the above example modularises (even if there is dependency between the variables) because one variable is repeating at $\frac{3}{2}$ times the frequency of the other. On the other hand, if we turn the ABC task into an $ABCD$ sequence, meaning the frequencies of the two tasks are related by a factor of 2, then correlations between the memories ($A \neq E$) can lead to mixing (though whether it will is

a more complex question).

We now apply these ideas to explain an interesting difference between two recent studies. Basu et al. (2021) and El-Gaby et al. (2024) trained rodents to visit 2 and 4, respectively, locations in order (and on repeat) to receive reward. This task structure was constant over different problems, even though the sequence of locations would change from problem to problem. El-Gaby et al. (2024) found structured subspaces in mPFC holding memories of goal locations (though with additional subspaces progressing the memory during walking periods, like the delays we saw earlier; see J. C. Whittington, William Dorrell, et al. (2025)) and an additional set of 'progress' neurons which code solely for progress towards each goal location. Curiously, Basu et al. (2021), while observing goal tuned neurons, did not observe pure 'progress' neurons but instead progress mixed with location. At first glance since, for both tasks, progress loops an integer number of times each loop of goal locations (i.e., they're harmonically related by a factor of 2 or 4), we might expect neurons to have mixed coding for goal location and progress. However this isn't observed in El-Gaby et al. (2024) but is in Basu et al. (2021)—why?

A critical difference between the two tasks is that Basu et al. (2021) uses a 1D grid world while El-Gaby et al. (2024) uses a 2D grid world. This means that the rodents in El-Gaby et al. (2024) will see all locations at all possible 'progress states' between goals, whereas the rodents in Basu et al. (2021) will only ever see the two outer locations when they are goal states (as it's a 1D track) and therefore at the end of their 'progress' between goals. This induces a range dependence between progress and goal location in Basu et al. (2021) but not in El-Gaby et al. (2024). Hence why we see modular coding of progress in El-Gaby et al. (2024) but not in Basu et al. (2021).

Indeed, we observe this in simulation, fig. 10.6, on a task that required predicting both a sequence of three memories ($ABCABCABC$... where A, B, & C 'locations' are sampled from a set of 'locations' for each problem) and a sequence of rewards where the third sequence element is rewarded and the others not (001001001...) (no memory here, just predict reward on every third step). These two sequences are harmonically related, so if we induce range correlations between reward and memories, such that some locations (A, B, or C) are never rewarded, then the encodings are mixed, and conversely memories and reward are range independent (all A, B, or C are sometimes rewarded and sometimes unrewarded) then we observe modular populations of neurons coding for goal location subspaces and 'progress' to reward.

10.8 Different Subspace Algorithms

So far we have described a subspace algorithm in which action dependent matrices shuffle memories between subspaces that are connected to one another according to the structure of the underlying task. In this case the action matrices correspond to actions taken on an underlying graph. This is a general formalism and in many cases is the simplest (uses the smallest set of actions), however it is not the only choice and therefore not the only plausible subspace algorithm. For some tasks, there exist other simple ways to associate 'actions' that also solve the task. We now highlight this for a simple task (J. Chen et al. (2024) which is a follow up to Y. Xie et al. (2022)), and show that another subspace algorithm resolves a critical discrepancy between the predictions of the previously presented subspace algorithm and neural data.

The discrepancy regards how inputs flow into subspaces. In J. Chen et al., 2024 (up to three) items are presented to a monkey in order and then must be recalled in sequence[1]. Neurally, when each observation is initially presented they enter an input subspace—a holding pen—from which they flow *directly* to their respective subspace whether it was the first, second, or third presented observation, fig. 10.7A. Each memory then stays in that subspace until the go cue arrives. This 'fixed subspace' model is very different to our earlier presentation in which the first memory moves from the first subspace and then conveyor belts into the second and third subspace, fig. 10.7B, but the 'fixed subspace' model can sometimes be learned by RNNs solving this task (M. Wang, Fusi, and K. Stachenfeld, 2025).

In order to understand this, let us consider the assumptions about actions in our original structured subspace model. Previously we had associated the action `Forward` to every transition. However, in the most general sense, we could have associated a unique action (and unique matrices, $\boldsymbol{W}_t$) at every timestep and this could still solve the task (this is because, in this task, every problem sequence has the same sequence of actions, i.e. actions aren't random; if they were random then unique matrices (and unique matrices, $\boldsymbol{W}_t$) at each timestep wouldn't be able to solve all problem sequences). This changes the subspace algorithm, and in a way that depend on which matrices are forced to be the same and which aren't. We note that, regardless of the subspace algorithm, there will always be an input subspace and a readout subspace as these are defined by the readin and readout weights respectively and not the recurrent matrices.

Assuming a RNN as before ($\boldsymbol{g}_t = \boldsymbol{W}_{\boldsymbol{a}_{t-1}}\boldsymbol{g}_{t-1} + \boldsymbol{W}_0\boldsymbol{o}_t\mathbf{1}_{\{\boldsymbol{s}_t \text{ unvisited}\}}$ with readout $\hat{\boldsymbol{o}}_{t+1} = \boldsymbol{R}\boldsymbol{W}_{\boldsymbol{a}_{t+1}}\boldsymbol{g}_t + \boldsymbol{b}$), let's consider the case where two items are presented and must be recalled in order. In this case there are 5 timesteps in the task (input 1, input 2, delay, recall 1, recall 2) and so 5 potential matrices, $\boldsymbol{W}_{\boldsymbol{a}_{t-1}}$. When the

[1]We ignore the reversing of sequences studied by J. Chen et al. (2024)

early matrices are the same ($\boldsymbol{a}_1 = \boldsymbol{a}_2 = \boldsymbol{a}_3$) then there must be a conveyor belt from the input subspace as before. But when the early matrices are all be different ($\boldsymbol{a}_1 \neq \boldsymbol{a}_2 \neq \boldsymbol{a}_3$), the input subspace can project to different storage subspaces on timestep 2 and timestep 3 (Figure **[James: XXX]**), effectively gating the flow of information from the input subspace to each subspace, and indeed this is what happens in simulation which matches the monkey data from J. Chen et al. (2024).

But how do memories get recalled after the delay period (assuming the first 3 matrices are all different from one another)? If the final matrices are the same ($\boldsymbol{a}_4 = \boldsymbol{a}_5$) then they must conveyor belt out, fig. 10.7D. However, if the final matrices are different from one another ($\boldsymbol{a}_4 \neq \boldsymbol{a}_5$) then the these two storage subspaces don't have to conveyor belt out, but the flow from each subspace can be gated at each timestep to the readout subspace, fig. 10.7C. J. Chen et al. (2024) does not report what happens post the delay nor does Y. Xie et al. (2022), however we can infer that the conveyor belt on the output is the likely option from the data in Y. Xie et al. (2022). Following our arguments presented earlier, subspace size is determined by 'adjacency' to the readout subspace and since we know that the subspaces in Y. Xie et al. (2022) have different sizes, this implies that they have different distances to the readout subspace which is only true for the output conveyor belt subspace structure, fig. 10.7.

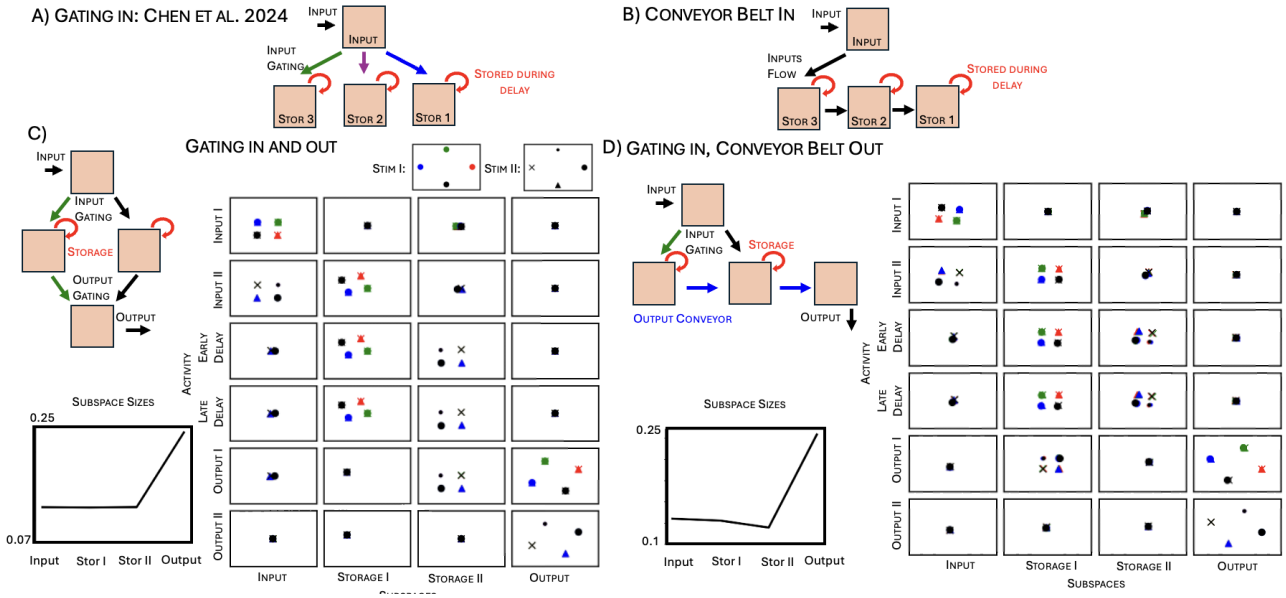


Figure 10.7: **A:** Chen et al. 2024 observe an input gating structure. Inputs first arrive in a holding-pen, the input subspace, and then are mapped to their corresponding storage subspace, the first stimulus to the first subspace, the second to the second, etc. **B:** This stands in contrast to our models thus far which have assumed a more 'conveyor belt' model, in which inputs flow through each of the preceeding subspaces on their way to the relevant storage subspace. By changing our choice of action matrices, we can implement a variety of subspace algorithms. **C:** We implement an input and output gating algorithm by providing the network with differnet input actions ('input1' and 'input2') and the same for output. In the plots we show the activity in each of the four identified subspaces (columns) at six different timepoints in the trial. It can be seen that the inputs briefly appear in the input subspace, before moving to their respective storage subspace and staying until mapping to the output. In the bottom left we show that the two storage subspaces are the same size, since they are equally close to recall. **D:** By giving the network only one 'output' action we force it to construct an output conveyor belt. In this case, the encoding of stimulus 2 is, as before, in the 2nd storage subspace during the delay period, but then reappears in the first storage subspace on its way to the output. This flow creates a hierarchy in distance from the output, causing the second storage subspace to be smaller than the first. Since this is observed in data, we conjecture that the monkey uses an output conveyor belt.

This results raise a question: Given the fully conveyor belt model is 'simpler', only requiring one matrix, why has the monkey brain chosen to use multiple input to storage subspaces transformation (and potentially uses a conveyor for the output)? We suggest two reasons. First, the monkeys were trained on sequences with a variable number of stimuli, between 1 and 3. This is clunky with a conveyor as even when the sequence only has one memory it has to be conveyed all the way round to be readout. This is not true for the fixed subspace model. On the other hand, this is only a problem for inputting the memories, but not for outputting them; once you have the fixed subspaces filled up then, regardless of the number of remembered stimuli, they can be simply conveyed to the readout subspace. This could be tested by training agents with sequence of a fixed length and seeing if they learn a conveyor belt. Second, the monkeys perform the output much faster than the input is

provided. Thus a simple model that requires no gating (output conveyor) might be preferred at high speeds. This could be tested by forcing the monkey to perform actions at a particular, slower, frequency.

The recent results of J. Chen et al. (2024), along with El-Gaby et al. (2024), Panichello and Buschman (2021), and Y. Xie et al. (2022), show that different subspace algorithms are taking place in PFC. In this section we have shown that the general principles of the ECH can be used to reason about their implementation, extending beyond our initial framing, with the critical differentiator being how transition matrices correspond to task variables.

10.9 Discussion

In this work we introduced a normative 'efficient computing' theory for thinking about the implementation of structured working memory representations. We formalised an optimisation problem in which memories are iteratively passed around a representation (the 'computable' constraint), before being recalled at the correct moment (the 'functional' constraint). We saw, section 10.3, that this led to subspaces for each memory a given action away from being recalled. Then, section 10.4, we reasoned about an efficient neural implementation of these subspaces. We saw that (range) correlated memories should be encoded in aligned subspaces, and used this to sketch out an axis of task-diversity which determined the degree of alignment, and matched this to neural data, section 10.5. Similarly, we considered the sizes of different subspaces, and saw this could be understood via an action-dependent 'distance to readout', that again matched data, section 10.6. We speculated about single neuron implementations of these algorithms, section 10.7, allowing us to match some, but not all, responses. And, finally, we considered alternative 'subspace algorithms', and used our theory to infer a putative algorithm from activity at only one timepoint section 10.8. We note one aspect that makes this final example especially pleasing. Y. Xie et al. (2022) recorded using calcium imaging, making the time resolution too slow to differentiate the two alternative algorithms. Yet, by reasoning about the subspace structure, we can make algorithmic inferences from limited data. We hope that this theory will prove similary useful for anyone reasoning about the neural implementation of these 'subspace algorithms'.

These algorithms are intriguing, by manipulating subspaces they neurally implement computations that can be applied flexibly, independent of the item encoded in the subspace. In this way, they reflect a neural implementation of a function: the same operation is applied independent of the argument - the activity within the subspace. We suspect that their flexibility will make them an important mode of neural computation, and we hope that our normative theory will provide helpful intuition and predictions for reasoning about neural recordings of these algorithms, in both brains and RNNs.

That said, our framework is clearly limited in a number of ways. Even within the limited scope of structured working memory problems, There are aspects of data that we do not fit. El-Gaby et al. (2024) demonstrate that their structured working memory representation lives on top of an encoding of task phase—every neuron is tuned to percentage progress towards goal, on top of which we observe the reported future-goal encoding. We are not currently able to normatively explain this. Similarly, our model predicts that orthogonal subspaces, as in Panichello and Buschman (2021), should be encoded in different neurons, but they are not. Broadly, it seems that our ability to fit single neuron data is still fairly limited. These are useful smoking guns, what are we missing in our model to explain these effects? Further, despite the simplicity of our approach, the maths was quite hard, meaning many of our results are limited, a situation that could likely be improved by better mathematicians than us.

More expansively, our framework is poorly placed to answer many naturally arising questions. What determines which subspace algorithm the brain uses when there are multiple available, section 10.8? How much flexibility with respect to inputs is required for a subspace algorithm to be preferable over alternatives? How are the shufflings of these subspaces combined with meaningful computations within the subspace, such as sensory-motor transformations (Stokes et al., 2013)? What determines when to 'chunk' the memory codes into hierarchically structured segments (Soni and Frank, 2025)? Finally, our framework sweeps the gating mechanism under the rug, simply assuming action-dependent matrices are available. This is likely to happen in local circuitry, indeed chapter 2 highlights how these effects can arise, and a more thorough theory would consider the costs and impacts of these choices.

Despite this long list of shortcomings, we found this process helpful. This simple theory can explain some otherwise puzzling parts of neural data, even allowing us to infer algorithmic structure from limited neural measurements. Further, the basic framework of this theory, the efficient computing hypothesis, is identical to one we developed to understand path-integrating representations, such as grid cells. We hope that ECH-type approaches, by framing interesting computations in mathematical tractable problems, will be useful in settings including representing arbitrary algorithmic processes, both hierarchical and recursive, and that this will offer explanation to areas of frontal cortex beyond motor and mPFC.

TECHNICAL APPENDICES: PREFRONTAL STRUCTURED WORKING MEMORY

10.A Mathematical Results

Here we detail the theory underpinning our main results. In particular, we formalise and study an optimal representation for storing structured memories, whose resulting characteristics match those in measured neural data from prefrontal cortex. To do this, we pose a natural optimisation problem of efficient memory storage within a recurrently connected population of biological neurons. Briefly, the problem is to exactly store memories while using as little neural activity as possible, as little synaptic activity as possible, and with non-negative neural activity. The optimal representations showcase (A) the existence of subspaces encoding different memories, the (B) alignments and (C) relative sizes of these subspaces, and (D) the corresponding single neuron representations.

Because the full optimisation problem is difficult to mathematically analyse, we instead restrict our analysis to two simplified settings. In the first setting, after modifying the loss on the recurrent weights and removing the nonnegativity constraint on neural activity, we find that the optimal representation is largely structured by the correlations between the encoded memories. In the second setting, after modifying the loss on the recurrent weights and the loss on neural activity, we find that, rather than correlations, it is range (in)dependence of the memories that governs subspace structure. Furthermore, these two simplified settings form a continuum between when subspace structure is determined by statistical (in)dependence or range (in)dependence. Finally, we use the intuition developed from these simple settings to understand and predict the behaviour of the loss without simplifications.

We therefore develop our approach as follows:

- In Section 10.A.1 we formalise the optimisation problem.
- In Section 10.A.2 we show that, in all versions of the problem, the representation is made up of subspaces encoding memories at each relative position (phenomenon A above).
- In Section 10.A.3 we rewrite the optimisation in terms of the subspace similarity structure, permitting further analysis. Further, we introduce the simplified, tractable version of the recurrent weight loss.

We then study the three problems of increasing complexity:

- In Section 10.A.4 we analyse the first simplified setting (the simplified recurrent weight loss and no nonnegative activity constraint). In this setting we understand and make proofs about the alignment of the subspaces (phenomenon B above). We show that independent memories are stored in orthogonal subspaces, while linearly correlated memories are stored in aligned subspaces.
- In Section 10.A.5 we study the second simplified setting (the simplified recurrent weight loss with simplified activity loss). In this setting we prove that all range-independent memories are optimally encoded in disjoint sets of neurons (phenomena B and D above), i.e. orthogonally. We discuss the continuum that these two simplified problems lie on.
- In Section 10.A.6 we use intuition from the previous solutions to propose ansatz solutions to two neuroscience relevant tasks, and show that the solutions match numerical simulations of the full optimisation problem.

Finally:

- In Section 10.A.7 we extract predictions based on qualitative trends that govern how various parameters in the problem effect the alignment and sizes of subspaces.

10.A.1 Problem Formalism

Task: Our problem is a memory problem: remembering events in the past; however, additionally, these events are structured. For examples, after moving around the physical world and observing what observation is at every location, you may be tasked with recalling which observation will be at each position. In that example, the underlying structure is physical 2D space, to which the memories are connected, though our formalism will apply to any underlying structure.

We formalise our task and representation as follows. Each state in the world, s, is associated with its own observation, $\boldsymbol{o}_e(s) \in \mathbb{R}^K$, e.g., this vector might encode the presence of a chair in location s. The underscore e denotes the current environment. We assume that the marginal distribution of observations in each state is

the same[2], is mean zero ($\langle \boldsymbol{o}_e(s) \rangle_e = \mathbf{0}$), and that the observations span the entire K dimensional space of their encoding.

Neural encoding: The neural encoding, $\boldsymbol{g} \in \mathbb{R}^N$, where N is the number of neurons, must store and appropriately recall these observations as one transitions in the world / underlying structure. That is, $\boldsymbol{g}$, after experiencing all observations, must be able to answer all counterfactual questions "what would I observe if I took the following action sequence?", even if that particular action sequence has not been taken before. Thus, $\boldsymbol{g}$ must be a function of both the set of observations it has encoded from the current environment, $\{\boldsymbol{o}_e(s), \forall s\}$, and the current state that you are in, s. We therefore write it as $\boldsymbol{g}(s, e)$.

Optimisation Constraints

Base Recall Constraint: First, $\boldsymbol{g}$ must be able to recall a memory. We formalise this by enforcing the decodability of the observation in the current position using a fixed affine function of the representation:

$$\boldsymbol{R}\boldsymbol{g}(s, e) + \boldsymbol{r} = \boldsymbol{o}_e(s) \tag{10.23}$$

Where $\boldsymbol{R} \in \mathbb{R}^{K \times N}$ and $\boldsymbol{r} \in \mathbb{R}^K$. This means that there are readout weights that map the representation at your current position to the observation that you expect to find there, satisfying the role of memory recall.

Structured Recall Constraint: Second, $\boldsymbol{g}$ must be able to recall memories not just related to your current position, but for *any* position. We implement this via a structured map constraint on $\boldsymbol{g}$ using affine functions. In particular, if there is an action a that takes you from state s to state $s + a$ then there must be a weight matrix and bias that does the same in neural space:

$$\boldsymbol{W}_a \boldsymbol{g}(s, e) + \boldsymbol{b}_a = \boldsymbol{g}(s + a, e) \tag{10.24}$$

Where $\boldsymbol{W}_a \in \mathbb{R}^{N \times N}$ and $\boldsymbol{b}_a \in \mathbb{R}^N$. This ensures that if you are at state s and you want to query what observation you will find after taking action a, you simply have to act on the representation first with the appropriate affine transform corresponding to 'take action a', then apply the readout transformation:

$$\boldsymbol{R}(\boldsymbol{W}_a \boldsymbol{g}(s, e) + \boldsymbol{b}_a) + \boldsymbol{r} = \boldsymbol{R}\boldsymbol{g}(s + a, e) + \boldsymbol{r} = \boldsymbol{o}_e(s + a) \tag{10.25}$$

Therefore, with an appropriate combination of action and readout transformations you are able to query this representation about the observation found in any part of the environment. Beyond being mathematically tractable, we justify the biological plausibility of these equations in Section 10.A.1.

Affine Read-in Constraint: We assume each memory in encoded when each observation is first observed using an affine read-in transformation. For example, starting a blank representation $\boldsymbol{g}_0 = \mathbf{0}$, then on observing $\boldsymbol{o}_e(s_1)$, it is encoded into the representation via:

$$\boldsymbol{g}_1 = \boldsymbol{W}_{\text{in}} \boldsymbol{o}_e(s_1) + \boldsymbol{b}_{\text{in}} \tag{10.26}$$

Where $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{N \times K}$ and $\boldsymbol{b}_{\text{in}} \in \mathbb{R}^N$. This process is repeated on first presentation of subsequent observations, after updating the representation appropriately with the map constraint, i.e.

$$\boldsymbol{g}(s, e) = \boldsymbol{W}_a \boldsymbol{g}(s - a, e) + \boldsymbol{b}_a + \boldsymbol{W}_{\text{in}} \boldsymbol{o}_e(s) + \boldsymbol{b}_{\text{in}} \tag{10.27}$$

We see that we are simply enforcing $\boldsymbol{g}$ to be a recurrent neural network (RNN). Notably, while 10.27 appears to show a linear RNN, in fact there are an implicit non-linearities, since we *only* read-in observation on first presentation, i.e., we are gating the input depending on whether that particular state has been visited before, and we gate the choice of recurrent weights depending on the action.

Nonnegative Constraint Finally, we constrain the representation to be non-negative, matching the nonnegative firing rates found in biological brain neurons:

$$\boldsymbol{g}(s, e) \geq 0 \quad \forall \quad s, e \tag{10.28}$$

Occasionally we drop this constraint as it can complicate the mathematical analysis.

Optimisation Objective

Subject to this set of constraints, we choose the representation that minimises the amount of energy used, either through neural activity, which we penalise for deviating from a target firing rate, t, or through large synapses (weight matrices). Our loss is:

$$\mathcal{L} = \lambda_A \sum_{s,e} ||\boldsymbol{g}(s, e) - t\mathbf{1}||^2 + \lambda_R ||\boldsymbol{R}||_F^2 + \lambda_W \sum_{a \in \mathcal{A}} ||\boldsymbol{W}_a||_F^2 \tag{10.29}$$

[2] i.e. the probabilities of each observation occurring in a given state over environments are the same for all states

Where λ_A, λ_R, and λ_W are regularisation hyperparameters that determine the relative strength of each term. We choose not to penalise the read-in weights for mathematical simplicity, though doing so does not qualitatively change the solutions. Further, depending on the problem being studied we might only penalise a subset of the weight matrices, specified by the set $\mathcal{A}$. This will be specified on a case-by-case basis. Finally, the recurrent weight loss is a challenge to analyse, so we will, at times, introduce a simplified version. This makes the whole problem analytically tractable, while preserving all the key phenomena in the solution as confirmed by numerical analysis.

In the subsequent sections we show that by purely optimising this loss under the constraints above, the resulting representations exhibit many neural phenomena observed in prefrontal cortex. We note that a very similar set of constraint (though without the requirement to encode memories in neural activity) gives rise to a mathematical formalism of entorhinal grid cells Will Dorrell et al., 2023. The critical difference between the grid representations and the prefrontal working memory representation is whether memories are stored in neural activity (prefrontal cortex), or in synaptic connections (hippocampus which can be queried by entorhinal cortex) J. C. Whittington, William Dorrell, et al., 2025.

Biological Plausibility of Action-Dependent Weight Matrices

Here we show that while our representational update mechanism (Equation (10.24)) may seem biologically implausible, it is in fact closely related to known attractor models of the brain.

The apparant implausibility come from the fact that each action acts on the population via a different affine transformation, which naively suggests that the synpases between neurons are changing strengths based on the current action. Fortunately models have been developed, originally to study thalamo-cortical interactions Logiaco, Abbott, and Escola, 2021 (but also matched to the fly ring attractor W. Zhang, Y. N. Wu, and S. Wu, 2022), that plausibly implement such an effective synaptic connectivity change. In this model a set of state encoding neurons are modelled as a linear RNN, bidirectionally and linearly connected to a second set of action-coding neurons (originally posed as the thalamus). These action-coding neurons are not recurrently connected amongst themselves, but they are gated on or off by the current choice of action. To make this precise, imagine there were two groups of action-coding neurons, $\boldsymbol{t}_a$ and $\boldsymbol{t}_{a'}$. Having chosen an action, one group, corresponding to the chosen action, will be free to evolve according to the linear dynamical system, while the second will be turned off. In maths, denote the state activity $\boldsymbol{g}$, the action activity $\boldsymbol{t}_a$ and $\boldsymbol{t}_{a'}$, the recurrent state connectivity $\boldsymbol{J}$, and the weights from $\boldsymbol{g}$ to each action-coding group $\boldsymbol{A}_a$ and $\boldsymbol{A}_{a'}$, and the weights from each action group back to $\boldsymbol{g}$ as $\boldsymbol{B}_a$ and $\boldsymbol{B}_{a'}$, then the linear dynamics become:

$$\dot{\boldsymbol{g}} = -\boldsymbol{g} + \boldsymbol{J}\boldsymbol{g} + \boldsymbol{B}_a\boldsymbol{t}_a + \boldsymbol{B}_{a'}\boldsymbol{t}_{a'} \qquad \text{State representation evolution} \qquad (10.30)$$

$$\dot{\boldsymbol{t}}_{\mathrm{a}} = -\boldsymbol{t}_{\mathrm{a}} + \boldsymbol{A}_a\boldsymbol{g} \qquad \text{Chosen action representation evolution} \qquad (10.31)$$

$$\boldsymbol{t}_{a'} = 0 \qquad \text{Not chosen action representation gated off} \qquad (10.32)$$

Because the action neurons are not themselves recurrently connected, their dynamics will settle to equilibrium much faster than the state dynamics. As such, we can assume $\boldsymbol{t}_a$ reaches it's steady state $\boldsymbol{t}_a = \boldsymbol{A}_{\mathrm{a}}\boldsymbol{g}$ before $\boldsymbol{g}$ changes significantly. The $\boldsymbol{g}$ dynamics then become:

$$\dot{\boldsymbol{g}} = -\boldsymbol{g} + \boldsymbol{J}\boldsymbol{g} + \boldsymbol{B}_a\boldsymbol{A}_{\mathrm{a}}\boldsymbol{g} = -\boldsymbol{g} + (\boldsymbol{J} + \boldsymbol{B}_a\boldsymbol{A}_a)\boldsymbol{g} \qquad (10.33)$$

Thus there are different effective recurrent connectivities in our population through the different weight matrices $(\boldsymbol{J} + \boldsymbol{B}_a\boldsymbol{A}_{\mathrm{a}})$. The choice of gating can then correspond to the current action. Thus using action dependent matrices (as in our optimisation problem) has an equivalence to how actions are believed to integrated in biological neural networks.

10.A.2 Activity Decomposes into (Linearly Independent) Memory Subspaces

Here we show that just assumptions of affine read-in (Equation (10.26)), readout (Equation (10.23)), and movement (Equation (10.24)) transformations together mean that the neural activity is be made up of a series of linear subspaces, each encoding the observation of a state that can be reached by a given action. For example, in a 2D grid world the neural activity will be made up of one subspace encoding the observation at your current state, another for the observation one step to the north, another for two steps north, another for one step east, etc. We now describe why this is the case.

Transformations without Bias

For now let us simply assume that the read-in, read-out, and recurrent transformations are linear rather than affine (i.e., the biases are all 0). The activity after one step will be:

$$\boldsymbol{g}_1 = \boldsymbol{W}_{\mathrm{in}}\boldsymbol{o}_e(s_0) \qquad (10.34)$$

Since the set of observation vectors $\{\boldsymbol{o}_e(s)\}$ span a K dimensional space, the encoded observation lives within a K dimensional subspace in neural activity spanned by the columns of $\boldsymbol{W}_{\text{in}}$.

After an action a, a second observation arrives, and the representation updates itself:

$$\boldsymbol{g}_2 = \boldsymbol{W}_a \boldsymbol{W}_{\text{in}} \boldsymbol{o}_e(s_0) + \boldsymbol{W}_{\text{in}} \boldsymbol{o}_e(s_1) \tag{10.35}$$

where $s_1 = s_0 + a$. Now the first memory has moved into a subspace spanned by the columns of $\boldsymbol{W}_a \boldsymbol{W}_{\text{in}}$, while the new memory $\boldsymbol{o}_e(s_1)$ lives in the subspace spanned by the columns of $\boldsymbol{W}_{\text{in}}$. To simplify notation let's define a new set of subspace matrices, $\boldsymbol{U}_0 = \boldsymbol{W}_{\text{in}}$, $\boldsymbol{U}_a = \boldsymbol{W}_a \boldsymbol{W}_{\text{in}}$, $\boldsymbol{U}_{2a} = \boldsymbol{W}_a \boldsymbol{W}_a \boldsymbol{W}_{\text{in}}$ and so on.

We see that as the agent travels from state to states encoding each observation into its working memory, $\boldsymbol{g}$, the memories are shuttled from subspace to subspace. For this to work correctly, the action matrices must obey certain constraints. For example if one action, a, is the inverse of another, a', meaning $(s + a) + a' = s$, then the corresponding action matrices must be inverses of one another: $\boldsymbol{W}_a = \boldsymbol{W}_{a'}^{-1}$, to ensure that the neural representation correctly keeps track of your state:

$$\boldsymbol{g}(s, e) = \boldsymbol{g}((s + a) + a', e) = \boldsymbol{W}_a \boldsymbol{W}_{a'} \boldsymbol{g}(s, e) = \boldsymbol{W}_a \boldsymbol{W}_a^{-1} \boldsymbol{g}(s, e) = \boldsymbol{g}(s, e) \tag{10.36}$$

This is just the structured recall constraint (Equation (10.24)) in action. In general the action matrices must match the rules of the environment. For example, if you are on a 2D grid world then moving north, then east, is the same as moving east, then north. Correspondingly, the action matrices associated with north and east must commute to ensure your representation is independent of route:

$$\boldsymbol{g}(s_{NE}, \boldsymbol{o}_e(s)) = \boldsymbol{W}_N \boldsymbol{W}_E \boldsymbol{g}(s, e) = \boldsymbol{W}_E \boldsymbol{W}_N \boldsymbol{g}(s, e) \qquad \boldsymbol{W}_N \boldsymbol{W}_E = \boldsymbol{W}_E \boldsymbol{W}_N \tag{10.37}$$

These consistency requirements on the action matrices tell us that in a working representation the action matrices will shuttle the information between the appropriate encoding subspaces.

Eventually, after all observations in the current environment have been loaded in, we have the full representation:

$$\boldsymbol{g}(s, e) = \sum_a \boldsymbol{U}_a \boldsymbol{o}_e(s + a) \tag{10.38}$$

There will be one subspace for each of the relative positions (defined by some action a away from the current position s) in the environment. Note that the actions a in this equation are not restricted to being one step away, but can be multiple (e.g., a' may correspond to 3 steps north and 1 steps east), whereas the actions in the recurrent weight matrices, $\boldsymbol{W}_a$, are for one step at a time.

Once the memory representation is fully constructed it takes no further observation inputs, simply updating itself according to the action matrices which has the effect of appropriately shuffling the observations around the memory representation. For example, imagine a world with two states, and correspondingly two subspaces:

$$\boldsymbol{g}(s, e) = \boldsymbol{U}_0 \boldsymbol{o}_e(s) + \boldsymbol{U}_a \boldsymbol{o}_e(s + a) \tag{10.39}$$

The only action available in this world is the one that swaps your state. Since swapping your state twice corresponds to staying where you are, the corresponding action matrix must square to identity: $\boldsymbol{W}_a^2 = \mathbb{1}$. Then if you take the only action available to you the representation updates:

$$\boldsymbol{W}_a \boldsymbol{g}(s, e) = \underbrace{\boldsymbol{W}_a \boldsymbol{U}_0}_{\boldsymbol{U}_a} \boldsymbol{o}_e(s) + \underbrace{\boldsymbol{W}_a \boldsymbol{U}_a}_{\boldsymbol{W}_a^2 \boldsymbol{U}_0 = \boldsymbol{U}_0} \boldsymbol{o}_e(s + a) = \boldsymbol{U}_0 \boldsymbol{o}_e(s + a) + \boldsymbol{U}_a \boldsymbol{o}_e(s) \tag{10.40}$$

And we can see that the observations swap subspaces.

Returning to the more general case, we consider the functional role these subspaces have to play. Independent of the behaviour of all other memories we have to be able to decode the current observation using our linear readout (Equation (10.23)):

$$\boldsymbol{R}\boldsymbol{g}(s, e) = \sum_a \boldsymbol{R}\boldsymbol{U}_a \boldsymbol{o}_e(s + a) = \boldsymbol{o}_e(s) \tag{10.41}$$

For this to be true the readout activity must perfectly extract the K dimensional subspace encoding the current memories: $\boldsymbol{R}\boldsymbol{U}_0 = \mathbb{1}$, while ignoring all the other subspaces: $\boldsymbol{R}\boldsymbol{U}_a = \boldsymbol{0}$. This is only possible if the K columns of $\boldsymbol{U}_0$ are linearly independent of the columns of each of the other subspaces $\boldsymbol{U}_a$, as otherwise information contained in the other subspaces would be read-out by the read-out matrix. We can extend this argument further, applying first an action matrix and then the readout matrix lets us make the same argument about a different subspace:

$$\boldsymbol{R}\boldsymbol{W}_{a'} \boldsymbol{g}(s, e) = \sum_a \boldsymbol{R}\boldsymbol{W}_{a'} \boldsymbol{U}_a \boldsymbol{o}_e(s + a) = \sum_a \boldsymbol{R}\boldsymbol{U}_{a+a'} \boldsymbol{o}_e(s + a) = \boldsymbol{o}_e(s - a') \tag{10.42}$$

Therefore we derive a new condition on the behaviour of $\boldsymbol{RW}_{a'}$; similarly to above it must extract the full K dimensional subspace encoded in subspace $\boldsymbol{U}_{-a'}$, $\boldsymbol{RW}_{a'}\boldsymbol{U}_{-a'} = \mathbb{1}$, while ignoring all other subspaces $\boldsymbol{RW}_{a'}\boldsymbol{U}_a = \boldsymbol{0}$. In order to do this, as above, the columns of $\boldsymbol{U}_{-a'}$ must be linearly independent, and linearly independent of the columns of all other subspaces.

Therefore we find that for the each of the N_A different actions, the representation needs to contain a K dimensional subspace, linearly independent from each of the others[3]. This subspace view matches the structures observed in PFC, and provides the framework in which our subsequent analysis proceeds.

Representation Decomposes even with Bias

Now we also show that the above ideas generalise to positive representations that require a bias, for example representations that are nonnegative. Thus we return the biases to the readin and recurrent transformations, i.e.:

$$\boldsymbol{g}_1 = \boldsymbol{W}_{\text{in}}\boldsymbol{o}_e(s_0) + \boldsymbol{b}_{\text{in}} \qquad \boldsymbol{g}(s+a, e) = \boldsymbol{W}_a\boldsymbol{g}(s, e) + \boldsymbol{b}_a \tag{10.43}$$

As actions are taken and new observations are seen, $\boldsymbol{g}$ now develop trailing bias terms:

$$\boldsymbol{g}_2 = \boldsymbol{W}_a\boldsymbol{W}_{\text{in}}\boldsymbol{o}_e(s_0) + \boldsymbol{W}_{\text{in}}\boldsymbol{o}_e(s_1) + \underbrace{\boldsymbol{b}_a + \boldsymbol{b}_{\text{in}} + \boldsymbol{W}_a\boldsymbol{b}_{\text{in}}}_{\boldsymbol{b}_2} \tag{10.44}$$

In general this bias vector will be a function of the sequence of actions taken. We develop our analysis of this trailing bias term in two separate settings: when the allowed actions do and don't form a group.

Transitions around space form a group: First, we consider the case where the set of allowed transitions form a group; for example, the map might be a periodic one or two dimensional space, Figure 10.8a. This tells us two things: first every subspace will always contain a memory, because from every point we can move using any action and expect to find something. Second, we can legitimately keep moving in any direction for ever, by applying the same transformation repeatedly. If the length of the bias vector is not kept constant by the recurrent affine transformations then eventually, as the trajectory lengthens, it will either explode (costing infinite activity energy) or go to 0, which isn't allowed because the representation must be positive and the only thing allowing this to be possible is the bias vector. Therefore, the length of the bias vector must be preserved under all recurrent transformations.

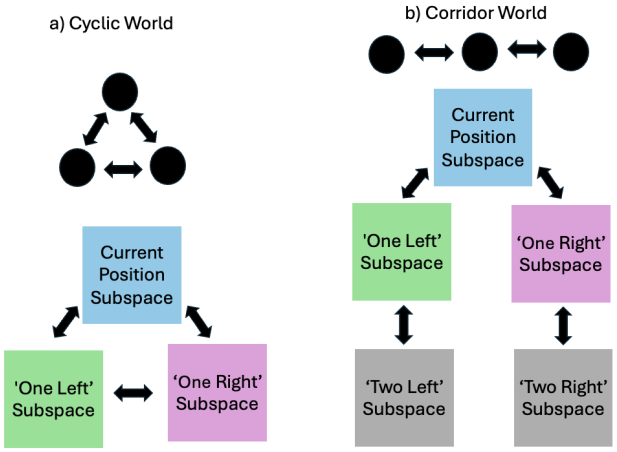


Figure 10.8: a) In a cyclic world like this you only need three subspaces, for observations at your current, left, and right positions. As you move each subspace will always contain a memory. b) However, in a corridor world you need more subspaces for the same number of states, and there will always be two empty ones.

Could the bias vector rotate though? The answer is no since, for any neuron, the bias should be as small as possible while still maintaining the nonnegativity of the representation, and rotating the bias vector increases and decreases the bias for each neuron thus costing energy. Furthermore, the bias should be action independent. This is because while there may be one environment where after taking action a there is less bias required in a given neuron, this will not happen across all environments since we have assumed that the marginal distributions of observations in each location are equal. Thus there will be another environment where the observations are exactly reversed, before taking action a a smaller bias is required, but after, a larger one. Therefore, in order

[3]This means there need to be at least KN_A neurons, else the network cannot solve the task

to ensure the representation is nonnegative in all environments while being optimally small, the bias must be action independent ($\boldsymbol{b}_a = \boldsymbol{b}$) and left unchanged by all affine action transformations: $\boldsymbol{W}_a\boldsymbol{b} + \boldsymbol{b}_a = \boldsymbol{b}$ for all a.

Now we consider reading out from a representation, and as in the previous section we will show that the subspaces are linearly independent.

$$\boldsymbol{R}\boldsymbol{g}(s,e) + \boldsymbol{r} = \boldsymbol{R}\sum_a \boldsymbol{U}_a\boldsymbol{o}_e(s+a) + \boldsymbol{R}\boldsymbol{b} + \boldsymbol{r} = \boldsymbol{R}\boldsymbol{U}_0\boldsymbol{o}_e(s) + \sum_{a\neq 0}\boldsymbol{R}\boldsymbol{U}_a\boldsymbol{o}_e(s+a) + \boldsymbol{R}\boldsymbol{b} + \boldsymbol{r} = \boldsymbol{o}_e(s) \tag{10.45}$$

In order for this decoding to happen correctly the readout matrix must faithfully copy the coding information in the current state subspace, $\boldsymbol{R}\boldsymbol{U}_0 = \mathbb{1}$ and must ignore all the other information, $\boldsymbol{R}\boldsymbol{W}_a = \boldsymbol{0}$. This is only possible if the columns of $\boldsymbol{U}_0$ are linearly independent from all the columns of the $\boldsymbol{U}_a$ matrices, as before (increasing $\boldsymbol{r}$ has no cost, so it doesn't matter whether $\boldsymbol{R}$ is orthogonal to $\boldsymbol{b}$, since we can simply choose $\boldsymbol{r} = -\boldsymbol{R}\boldsymbol{b}$). And, as we did in the previous section, we can extend the argument by considering reading out from another subspace by first taking an action, then reading out, $\boldsymbol{R}\boldsymbol{W}_a$, we derive again that the columns of all the subspaces need to be linearly independent.

Transitions do not form a group: Things are slightly more complex when the transitions do not form a group. For example, you might be in a one-dimensional corridor of length 3, Figure 10.8b. If you are in the middle subspace then the 'current position', 'one left', and 'one right' subspace will all have encoded memories. On the other hand, if you then move to the right, a different set of three subspaces, the 'current position', 'one left', and 'two left' subspaces will contain activity. Further, taking the 'move right' action from the furthest right point has no meaning, as that's the end of the corridor. Thus since all states are not equal, it may be optimal to change the length of the bias vector for each state. However, it will never be optimal to make it path dependent, since the aim of the bias (making the representation minimally-positive) is invariant to path used to reach a given state.

Therefore we end up with the following representation:

$$\boldsymbol{g}(s,e) = \sum_a \boldsymbol{U}_a\boldsymbol{o}_e(s+a) + \boldsymbol{b}(s) \tag{10.46}$$

However, this does not fundamentally change the claims. In order to correctly decode the subspaces their columns must be mutually linearly independent. The only slight complication is that the representational bias, $\boldsymbol{b}(s)$, and the readout bias, $\boldsymbol{r}$, have to be chosen so that they always cancel during readout:

$$\boldsymbol{R}\boldsymbol{g}(s,e) + \boldsymbol{r} = \sum_a \boldsymbol{R}\boldsymbol{U}_a\boldsymbol{o}_e(s+a) + \boldsymbol{R}\boldsymbol{b}(s) + \boldsymbol{r} = \boldsymbol{o}_e(s) \qquad \boldsymbol{R}\boldsymbol{b}(s) + \boldsymbol{r} = \boldsymbol{0}\forall s \tag{10.47}$$

This is a minor nuisance, and so, henceforth, we restrict to settings in which the transitions form a group. However we conjecture, and have empirical evidence, that our main results will generalise to cases where the transition structure is not a group.

Summary

In order to satisfy the constraints the representation must be made of linear subspaces encoding the observations at each relative position from the agent. If this is true there exist affine action maps ($\boldsymbol{W}_a$), read-in, and read-out transformations that can construct and decode the representation. We can therefore optimise the loss (Equation (10.28)) directly over the subspace structure subject to the remaining positivity constraint.

10.A.3 Rewriting Optimisation Problem in Terms of Subspace Structure

The previous section outlined how, as long as the representation has linearly independent subspaces, it will satisfy the constraints set forth in Equation (10.23), Equation (10.24), and Equation (10.26). Having satisfied these constraints, we seek the optimal energy efficient representation. Since $\boldsymbol{g}$ consists of linearly independent subspaces, it is convenient to rewrite the loss (Equation (10.28)) in terms of subspace structure. Later, we will simplify this loss in order to make it analytically tractable.

Analysing Losses

Some Bookkeeping: We are optimising the representation $\boldsymbol{g}$ in every state and all environments, and so let us stack all these vectors into a large matrix:

$$\boldsymbol{G} = \begin{bmatrix}\boldsymbol{g}(s_0,e_0) & \boldsymbol{g}(s_1,e_0) & \dots & \boldsymbol{g}(s_{N_s},e_0) & \boldsymbol{g}(s_0,e_1) & \dots & \boldsymbol{g}(s_{N_s},e_{N_e})\end{bmatrix} \in \mathbb{R}^{N\times N_S N_E} \tag{10.48}$$

where N_E and N_S are the number of environments, and the number of states each in each environment.

Each column of $\boldsymbol{G}$ corresponds to the shifted neural representation in a particular state, and can be written in its subspace decomposition.

$$\boldsymbol{G}_{:,iN_S+j} = \sum_a \boldsymbol{U}_a \boldsymbol{o}(s_j + a, e_i) + \boldsymbol{b}\mathbf{1}^T \tag{10.49}$$

To make the notation easier let's turn this into a single matrix vector equation with the following definitions:

$$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{U}_0 & \boldsymbol{U}_{a_1} & \dots & \boldsymbol{U}_{a_{N_A-1}}\end{bmatrix} \in \mathbb{R}^{N \times KN_A} \qquad \boldsymbol{o}_{iN_E+j} = \begin{bmatrix}\boldsymbol{o}_{e_i}(s_j) \\ \boldsymbol{o}_{e_i}(s_j + a_1) \\ \vdots \\ \boldsymbol{o}_{e_i}(s_j + a_{N_A-1})\end{bmatrix} \in \mathbb{R}^{KN_A} \tag{10.50}$$

Then simply $\boldsymbol{G}_{:,iN_E+j} = \boldsymbol{U}\boldsymbol{o}_{iN_S+j} + \boldsymbol{b}$. Further we can stack up all the observation vectors from the $N_S N_E$ different conditions into one matrix $\boldsymbol{O}$:

$$\boldsymbol{O} = \begin{bmatrix}\boldsymbol{o}_1 & \dots & \boldsymbol{o}_{N_S N_E}\end{bmatrix} \in \mathbb{R}^{KN_A \times N_S N_E} \tag{10.51}$$

Then even more simply $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{O} + \boldsymbol{b}\mathbf{1}^T$. Further, we will find that a convenient variable is the subspace dot product structure: $\boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{U} \in \mathbb{R}^{KN_A \times KN_A}$. We thus write each of the losses in terms of this subspace dot product matrix.

Activity Loss: The activity loss measures deviation of the firing rate from a target firing rate:

$$\begin{aligned}\sum_{s,e} ||\boldsymbol{g}(s,e) - t\mathbf{1}||^2 &= \text{Tr}[(\boldsymbol{G} - t\mathbf{1}\mathbf{1}^T)^T(\boldsymbol{G} - t\mathbf{1}\mathbf{1}^T)] \\ &= \text{Tr}\left[\begin{bmatrix}\boldsymbol{U}^T\boldsymbol{U} & \boldsymbol{U}^T(\boldsymbol{b} - t\mathbf{1}) \\ (\boldsymbol{b} - t\mathbf{1})^T\boldsymbol{U} & ||\boldsymbol{b} - t\mathbf{1}||^2\end{bmatrix}\begin{bmatrix}\boldsymbol{O}\boldsymbol{O}^T & \mathbf{0} \\ \mathbf{0} & N_S N_E\end{bmatrix}\right] \\ &= \text{Tr}[\boldsymbol{D}\boldsymbol{O}\boldsymbol{O}^T] + N_S N_E||\boldsymbol{b} - t\mathbf{1}||^2\end{aligned} \tag{10.52}$$

where we used the fact the observations are mean-zero, therefore $\boldsymbol{O}\mathbf{1} = 0$.

Readout Loss: Next, the readout matrix must simply extract the current subspace. For a given subspace structure the minimal L2 norm matrix that does this is given by the first K rows of the pseudoinverse matrix of $\boldsymbol{U}$ (because the first K rows of the pseudo inverse 'attend' to the first K columns of $\boldsymbol{U}$ which is the first subspace) , denoted by $\boldsymbol{U}_0^\dagger$, hence:

$$\text{Tr}[\boldsymbol{R}^T\boldsymbol{R}] = \text{Tr}[\boldsymbol{U}_0^{\dagger,T}\boldsymbol{U}_0^\dagger] \tag{10.53}$$

Let's define $\boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{U}$, which is the subspace dot product matrix without the bias. Then, by the pseudoinverse or otherwise, one can find that $\boldsymbol{U}^\dagger\boldsymbol{U}^{\dagger,T} = \boldsymbol{D}^{-1}$. Hence:

$$\text{Tr}[\boldsymbol{R}^T\boldsymbol{R}] = \text{Tr}[\boldsymbol{D}_{00}^{-1}] \tag{10.54}$$

Where the subscript 00 denotes the first $K \times K$ dimensional block of the matrix $\boldsymbol{D}^{-1}$.

Recurrent Loss: Last we have to consider the recurrent weight loss. Each recurrent weight matrix has to take the activity in one subspace and move it to its appropriate target subspace. For a given subspace structure the minimal L2 norm matrix that does this is the following:

$$\boldsymbol{W}_a = \sum_s \boldsymbol{U}_{s+a}\boldsymbol{U}_s^\dagger \tag{10.55}$$

The matrix decomposes activity onto axis aligned subspaces using the psuedoinverse matrices, $\boldsymbol{U}_s^\dagger$, then projects them back using $\boldsymbol{U}_{s+a}$. We can rewrite this using a permutation matrix, $\boldsymbol{P}_a$, that maps each K-dimensional block s to the block $s + a$: =

$$\boldsymbol{W}_a = \boldsymbol{U}\boldsymbol{P}_a\boldsymbol{U}^\dagger \tag{10.56}$$

Then the weight loss is:

$$\sum_{a\in\mathcal{A}} ||\boldsymbol{W}_a||_F^2 = \sum_{a\in\mathcal{A}} \text{Tr}[\boldsymbol{W}_a^T\boldsymbol{W}_a] = \sum_{a\in\mathcal{A}} \text{Tr}[\boldsymbol{P}_a^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{P}_a\boldsymbol{U}^\dagger\boldsymbol{U}^{\dagger,T}] = \sum_{a\in\mathcal{A}} \text{Tr}[\boldsymbol{P}_a^T\boldsymbol{D}\boldsymbol{P}_a\boldsymbol{D}^{-1}] \tag{10.57}$$

Combination: We can write the whole loss in terms of the subspace, and subspace-and-bias dot product matrices and problem specific parameters:

$$\mathcal{L} = \lambda_A(\text{Tr}[\boldsymbol{D}\boldsymbol{O}\boldsymbol{O}^T] + ||\boldsymbol{b} - t\mathbf{1}||^2) + \lambda_R \text{Tr}[\boldsymbol{D}_{00}^{-1}] + \lambda_W \sum_{a\in\mathcal{A}} \text{Tr}[\boldsymbol{P}_a^T\boldsymbol{D}\boldsymbol{P}_a\boldsymbol{D}^{-1}] \tag{10.58}$$

And this has to be minimised over $\boldsymbol{D}$ and $\boldsymbol{b}$, subject to the representation maintaining nonnegativity and linearly independent subspaces.

Non-Convexity: Deriving analytic results for the optimal solution to the above equation is difficult because it is non-convex. The term that causes the problems is the recurrent weight loss. We provide code that shows that the recurrent term is a nonconvex function of the subspace similarity matrix, $\boldsymbol{D}$[4].

Strategy: Given the difficulty in analysing the recurrent term we introduce a slightly simplified form of the recurrent weight loss. We then study the properties of this problem. We use these findings to propose ansatz (guess) solutions to the whole problem in a few cases of interest and show they match empirical findings.

Recurrent Weight Loss Simplification

Each recurrent weight matrix can be written as a sum of components (note this is the same as Equation (10.56) but where the permutation matrix is effectively performed by the sum over subspaces s):

$$\boldsymbol{W}_a = \sum_s \boldsymbol{U}_{s+a}\boldsymbol{U}_s^\dagger \tag{10.59}$$

Each component extracts the activity from one subspace using $\boldsymbol{U}_s^\dagger$, and projects it to the appropriate part of the representation with $\boldsymbol{U}_{s+a}$. The Frobenius norm can then be written:

$$||\boldsymbol{W}_a||_F^2 = \text{Tr}[\boldsymbol{W}_a^T\boldsymbol{W}_a] = \sum_{s,s'} \text{Tr}[\boldsymbol{U}_{s'+a}^T\boldsymbol{U}_{s+a}\boldsymbol{U}_s^\dagger\boldsymbol{U}_{s'}^{\dagger,T}] = \sum_{s,s'} \text{Tr}[\boldsymbol{D}_{s'+a,s+a}\boldsymbol{D}_{s,s'}^{-1}] \tag{10.60}$$

where the notation D_{ij} denotes the ijth K dimensional block of D. The difficulty in analysing this loss comes from the cross-terms[5]. We therefore introduce a simplified form of the loss which does not penalise these cross terms; rather, we penalising each section of the weight matrix separately:

$$\mathcal{L}_{\text{recurrent, simple}} = \sum_a\sum_s ||\boldsymbol{U}_{s+a}\boldsymbol{U}_s^\dagger||_F^2 = \sum_{a,s} \text{Tr}[\boldsymbol{U}_{s+a}^T\boldsymbol{U}_{s+a}\boldsymbol{U}_s^\dagger\boldsymbol{U}_s^{\dagger,T}||_F^2 = \sum_{a,s} \text{Tr}[\boldsymbol{D}_{s+a,s+a}\boldsymbol{D}_{s,s}^{-1}] \tag{10.61}$$

This version of the loss can be interpreted as if the movement from each subspace to another was implemented by a separate set of neurons, of the sort of 'action neurons' described in Section 10.A.1. This seems quite plausible, especially if the agent needs to combinatorially combine all possible subspace-to-subspace transitions, having them independently represented might confer significant advantages. Regardless, this seems a reasonable loss to study.

Therefore we arrive at our final loss function:

$$\mathcal{L}_{\text{simple}} = \lambda_A \underbrace{(\text{Tr}[\boldsymbol{D}\boldsymbol{O}\boldsymbol{O}^T] + ||\boldsymbol{b} - t\boldsymbol{1}||^2)}_{\text{activity}} + \lambda_R \underbrace{\text{Tr}[\boldsymbol{D}_{00}^{-1}]}_{\text{read-out}} + \lambda_W \underbrace{\sum_{a,s} \text{Tr}[\boldsymbol{D}_{s+a,s+a}\boldsymbol{D}_{s,s}^{-1}]}_{\text{recurrent}} \tag{10.62}$$

$$\tag{10.63}$$

Which we can rewrite a more simply with the following notation for the block-diagonalised and shifted-block-diagonalised matrices:

$$\boldsymbol{D}_B = \begin{bmatrix} \boldsymbol{D}_{00} & \boldsymbol{0} & ... \\ \boldsymbol{0} & \boldsymbol{D}_{11} & ... \\ \vdots & \vdots & \ddots \end{bmatrix} \qquad \boldsymbol{D}_{B_a} = \begin{bmatrix} \boldsymbol{D}_{aa} & \boldsymbol{0} & ... \\ \boldsymbol{0} & \boldsymbol{D}_{1+a,1+a} & ... \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{10.64}$$

After which we arrive at our final loss function:

$$\mathcal{L}_{\text{simple}} = \lambda_A(\text{Tr}[\boldsymbol{D}\boldsymbol{O}\boldsymbol{O}^T] + ||\boldsymbol{b} - t\boldsymbol{1}||^2) + \lambda_R \text{Tr}[\boldsymbol{D}_{00}^{-1}] + \lambda_W \sum_a \text{Tr}[\boldsymbol{D}_{B_a}\boldsymbol{D}_B^{-1}] \tag{10.65}$$

10.A.4 Solution I: Without Nonnegativity Constraint

Rough Trends

Without the nonnegativity constraint the bias plays no useful role, so it may be set to its minimal value, $\boldsymbol{b} = t\boldsymbol{1}$. We can then take the derivative of the loss with respect to $\boldsymbol{D}$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}} = \lambda_A \boldsymbol{O}\boldsymbol{O}^T - \lambda_R \boldsymbol{D}^{-1}\mathbb{1}_{00}\boldsymbol{D}^{-1} + \lambda_W \sum_a (\boldsymbol{D}_B^{-1} - \boldsymbol{D}_B^{-1}\boldsymbol{D}_{B_a}\boldsymbol{D}_B^{-1}) = \boldsymbol{0} \tag{10.66}$$

[4]The code shows it is also neither log-convex nor log-log-convex.

[5]For example, the proof in section 10.A.5 does not hold when these terms are included.

where $\mathbb{1}_{00}$ is an identity matrix only in the first K by K block. From this we can derive some interesting limiting behaviour:

1. If λ_W is much smaller than the other hyperparameters then $\boldsymbol{D} \approx \sqrt{\frac{\lambda_R}{\lambda_A}}\mathbb{1}_{00}(\boldsymbol{O}\boldsymbol{O}^T)^{-\frac{1}{2}}$. Thus the only meaningfully sized subspace is the readout one, whose dot product structure is determined by the 00th block of the observation dot product matrix. If λ_R is very big you want a large readout subspace so that the weight matrix can be small, while if λ_A is very big you want it to be small, and to instead grow the size of the readout weight matrix. This is all intuitive: if λ_W is negligible then very large recurrent weights can move content between subspaces in a way that ensures there is little activity used in all subspaces but the readout one (you need large recurrent weights from adjacent subspaces to the readout subspace). This ensures that the entire subspace structure is essentially driven by the only meaningfully sized subspace, the readout subspace, whose size has to be large due to the nonnegligble λ_R hyperparameter. Further, if the activity in the readout subspace also became very small (due to high λ_R) you would require very large readout weights to map the activity to the outputs.
2. If λ_W is much bigger than the other hyperparameters then $\boldsymbol{D} \approx \mathbb{1}$ is a solution, i.e. each subspace is orthogonal and equally sized to all other subspaces. This is again intuitive: the recurrent weight matrices have to shunt activity from one subspace to another, which requires both large and small weights if the subspaces are aligned or have different sizes. Large weights cost energy. If the subspaces are orthogonal and equally sized, then the recurrent weight matrix can also be orthogonal which is its lowest energy form.
3. The subspace structure is in part determined by $\boldsymbol{O}\boldsymbol{O}^T$ (from the activity loss), which describes the correlations of the observations. Thus introducing correlations in $\boldsymbol{O}\boldsymbol{O}^T$ will introduce correlations (alignment in neural space) between the subspaces.

Independent Memories are Orthogonalised, Correlated are Aligned

Let's start with the simplest case; when $\boldsymbol{O}\boldsymbol{O}^T$ takes a particularly simple block diagonal form. This occurs when the memories held in different subspaces are independent of one-another. Here we show that, in this case, the activity loss is independent of the off-diagonal blocks of $\boldsymbol{D}$ and that both weight losses are minimised by orthogonalising the slots. This means that the optimal representation for independent memories is when the subspaces are orthogonal to one-another.

Activity Loss

It is easy to see that $\boldsymbol{O}\boldsymbol{O}^T$ is block-diagonal for independent memories if we consider its ijth K dimensional block:

$$[\boldsymbol{O}\boldsymbol{O}^T]_{ij} = \sum_{s,e} \boldsymbol{o}_e(s+a_i)\boldsymbol{o}_e^T(s+a_j) = N_S N_E \langle \boldsymbol{o}_e(s+a_i)\boldsymbol{o}_e^T(s+a_j)\rangle_{e,a} \tag{10.67}$$

Across actions and environments the memories are independent and mean zero so this matrix is block diagonal, and the diagonal blocks are the identical covariance matrix of the observations in a single subspace:

$$\boldsymbol{O}\boldsymbol{O}^T = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} & \dots \\ \boldsymbol{0} & \boldsymbol{\Sigma} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad \boldsymbol{\Sigma} = N_A N_E \langle \boldsymbol{o}_e(s+a)\boldsymbol{o}_e^T(s+a)\rangle_{e,a} \tag{10.68}$$

This means that the activity loss only depends on the diagonal blocks of $\boldsymbol{D}$:

$$\mathrm{Tr}[\boldsymbol{D}\boldsymbol{O}\boldsymbol{O}^T] = \sum_a \mathrm{Tr}[\boldsymbol{D}_{aa}\boldsymbol{\Sigma}] \tag{10.69}$$

so is not affected by orthogonalisation, which only changes the off-block-diagonal parts of $\boldsymbol{D}$.

Readout Loss

We now consider the readout loss, and observe that the loss is minimised when subspaces are orthogonal.

The min-norm readout matrix is equal to the pseudoinverse of the output subspace encoding, $\boldsymbol{U}_0$: $\boldsymbol{R} = \boldsymbol{U}_0^\dagger$. In general, $\boldsymbol{U}_0^\dagger$ can be thought of as doing two jobs. First, within the first subspace (the space spanned by the columns of $\boldsymbol{U}_0$) its behaviour is determined undoes the subspace embedding for the first subspace, so that for any vector $\boldsymbol{x}$: $\boldsymbol{U}_0^\dagger \boldsymbol{U}_0 \boldsymbol{x} = \boldsymbol{x}$. Second, it ignores embe

can be broken into components, one that lies within the span of the columns of $\boldsymbol{U}_0$, and another orthogonal to all the columns:

$$\boldsymbol{U}_0^\dagger = \boldsymbol{U}_{0,\parallel}^\dagger + \boldsymbol{U}_{0,\perp}^\dagger \qquad ||\boldsymbol{R}||_F^2 = ||(\boldsymbol{U}_{0,\parallel}^\dagger + \boldsymbol{U}_{0,\perp}^\dagger)||_F^2 = ||\boldsymbol{U}_{0,\parallel}^\dagger||_F^2 + ||\boldsymbol{U}_{0,\perp}^\dagger||_F^2 \tag{10.70}$$

$\boldsymbol{U}_{0,\|}^{\dagger}$ is given by the condition that:

$$\boldsymbol{U}_0^{\dagger}\boldsymbol{U}_0 = (\boldsymbol{U}_{0,\|}^{\dagger} + \boldsymbol{U}_{0,\perp}^{\dagger})\boldsymbol{U}_0 = \boldsymbol{U}_{0,\|}^{\dagger}\boldsymbol{U}_0 = \mathbb{1}_{\text{span}} \tag{10.71}$$

Where $\mathbb{1}_{\text{span}}$ denotes identity within the span of the columns of $\boldsymbol{U}_0$ and zero elsewhere. In the modularisation procedure the within-subspace dot product matrix, $\boldsymbol{U}_0^T\boldsymbol{U}_0$, stays fixed. Further, the Frobenius norm of the parallel component of the pseudoinverse is actually then fixed, which can be seen from, among other things, the reduced SVD:

$$\boldsymbol{U}_0 = \boldsymbol{A}\boldsymbol{S}\boldsymbol{B}^T \qquad \text{Tr}[\boldsymbol{U}_0^T\boldsymbol{U}_0] = \text{Tr}[\boldsymbol{S}^2] = \sum_i S_{ii}^2 \qquad \boldsymbol{U}_0^{\dagger} = \boldsymbol{B}\boldsymbol{S}^{-1}\boldsymbol{A}^T \qquad \text{Tr}[\boldsymbol{U}_{0,\|}^{\dagger,T}\boldsymbol{U}_{0,\|}^{\dagger}] = \text{Tr}[\boldsymbol{S}^{-2}] = \sum_i S_{ii}^{-2} \tag{10.72}$$

So we just need to consider the perpendicular component. This only exists to ensure that the pseudoinverse is orthogonal to all other subspaces:

$$\boldsymbol{U}_0^{\dagger}\boldsymbol{U}_i = (\boldsymbol{U}_{0,\|}^{\dagger} + \boldsymbol{U}_{0,\perp}^{\dagger})\boldsymbol{U}_i = \boldsymbol{0} \qquad \boldsymbol{U}_{0,\|}^{\dagger}\boldsymbol{U}_i = -\boldsymbol{U}_{0,\perp}^{\dagger}\boldsymbol{U}_i \tag{10.73}$$

If the subspaces are made orthogonal the perpendicular component, $\boldsymbol{U}_{0,\perp}^{\dagger}$, is set to zero, necessarily reducing or, in the case that the $\boldsymbol{U}_0$ was already orthogonal to all the other subspaces, leaving invariant the readout weight loss.

Recurrent Loss

We now consider the recurrent loss, and observe that (due to a similar analysis to the above section) the loss is minimised when subspaces are orthogonal:

$$\sum_{a,s} \text{Tr}[\boldsymbol{D}_{s+a,s+a}\boldsymbol{D}_{s,s}^{-1}] = \sum_{a,s} \text{Tr}[\boldsymbol{U}_{s+a}^T\boldsymbol{U}_{s+a}\boldsymbol{U}_s^{\dagger}\boldsymbol{U}_s^{\dagger,T}] \tag{10.74}$$

If the subspaces are orthogonal, then $\boldsymbol{U}_{s+a}^T\boldsymbol{U}_{s+a}$ is constant. Thus we just need to consider the pseudoinverse terms. In a similar approach to the above section, $\boldsymbol{U}_i^{\dagger}$, can be broken into a component within the span of its subspace, $\boldsymbol{U}_i$, and one orthogonal to it, such that:

$$\text{Tr}[\boldsymbol{U}_{s+a}^T\boldsymbol{U}_{s+a}\boldsymbol{U}_s^{\dagger}\boldsymbol{U}_s^{\dagger,T}] = \text{Tr}[\boldsymbol{U}_{s+a}^T\boldsymbol{U}_{s+a}\boldsymbol{U}_{s,\|}^{\dagger}\boldsymbol{U}_{s,\|}^{\dagger,T}] + \text{Tr}[\boldsymbol{U}_{s+a}^T\boldsymbol{U}_{s+a}\boldsymbol{U}_{s,\perp}^{\dagger}\boldsymbol{U}_{s,\perp}^{\dagger,T}] \tag{10.75}$$

If the subspaces, $\boldsymbol{U}_s$, are orthogonal, then all perpendicular parts, $\boldsymbol{U}_{s,\perp}$, are zero while all the parallel parts are fixed. Thus orthogonal subspaces necessarily reduces this loss.

Correlated Memories

Conversely, if the memories are linearly correlated across states, i.e. the matrix $\boldsymbol{O}\boldsymbol{O}^T$ is no longer block diagonal, we can see that the subspace dot product matrix, $\boldsymbol{D}$ won't be block diagonal either. The activity loss will reduce if we vary the off-block-diagonal parts of $\boldsymbol{D}$ appropriately such that the off diagonal terms, such as $\text{Tr}[\boldsymbol{D}_{01}(\boldsymbol{O}\boldsymbol{O}^T)_{01}]$ are negative. Since the weight loss is minimised by orthogonal subspaces, introducing a small part of this off-block-diagonal component won't change the weight loss too much (to first order). Therefore linearly correlated memories will align subspacesat least a bit.

10.A.5 Solution II: Simplified Nonnegativity Problem with no Target Rate

In this section we study optimum representations, $\boldsymbol{g}$, for the loss with the simplified weight loss (Equation (10.65)), though we keep the nonnegativity constraint on neural firing, but drop the target firing rate ($t = 0$). We will show that, in this setting, range independent memories—this is when all combinations of memories in subspaces are allowed but statistical dependencies may exist, i.e. the range is independent—are optimally stored in different neurons. In this setting the only appearance of the bias in the loss is in the term $||\boldsymbol{b}||^2$, so the bias should be as small as possible, while guaranteeing the nonnegativity of the representation.

Following from a combination of our previous work, William Dorrell, K. Hsu, et al., 2025, and the previous results on minimising we consider an operation in which we break any neuron tuned to multiple memories apart. This operation modularises the encoding of different subspaces, therefore, as in Section 10.A.4, it necessarily reduces the weight losses. We show that for range-independent memories[6] this neuron modularising operation

[6] if two variables are statistically independent then knowing the value of one variable doesn't change the distribution of the other. Range independence means knowing the value of one variable doesn't change the range of possible values, or the support, of the other variable.

also necessarily reduces the activity loss, and therefore that the optimal representation only has neurons tuned to single memories.

Let's say we have a representation that contains neurons tuned to multiple memories, e.g.:

$$g_n(s,e) = \sum_a \boldsymbol{U}_{a,n}^T \boldsymbol{o}(s+a,e) + b_n = \sum_a \boldsymbol{U}_{a,n}^T \boldsymbol{o}(s+a,e) - \min_{s,e}[\sum_a \boldsymbol{U}_{a,n}^T \boldsymbol{o}(s'+a,e')] \geq 0 \tag{10.76}$$

where $\boldsymbol{U}_{a,n}$ is the nth row of the $\boldsymbol{U}_a$ subspace matrix. Now let's imagine breaking this into N_A separate, minimally nonnegative, neural tunings:

$$g_{n,a}(s,e) = \boldsymbol{U}_{a,n}^T \boldsymbol{o}(s+a,e) - \min_{s',e'}[\boldsymbol{U}_{a,n}^T \boldsymbol{o}(s'+a,e')] = \boldsymbol{U}_{a,n}^T \boldsymbol{o}(s+a,e) + b_{a,n} \geq 0 \tag{10.77}$$

Further, we use the range independence to break apart the bias term[7]:

$$\min_{s',e'}[\sum_a \boldsymbol{U}_{a,n}^T \boldsymbol{o}(s'+a,e')] = \sum_a \min_{s',e'}[\boldsymbol{U}_{a,n}^T \boldsymbol{o}(s'+a,e')] = \sum_a b_{a,n} \tag{10.78}$$

So we can actually write the neuron tuned to multiple memories as:

$$g_n(s,e) = \sum_a g_{n,a}(s,e) \tag{10.79}$$

Now we can see that this operation necessarily reduces the activity loss. This is because the modular cost is $\sum_a \langle (g_{n,a}(s,e))^2 \rangle_{s,e}$, while the mixed neural tuning can be written as the sum of the same modular cost and a postive term:

$$\langle g_n(s,e)^2 \rangle_{s,e} = \sum_a \langle (g_{n,a}(s,e))^2 \rangle_{s,e} + \sum_{a,a' \neq a} \langle g_{n,a}(s,e) g_{n,a'}(s,e) \rangle_{s,e} \tag{10.80}$$

And since the variables are range independent the inequality $\langle g_{n,a}(s,e) g_{n,a'}(s,e) \rangle_{s,e} > 0$ is strictly true. Since all losses are either reduced or remain constant in doing this modularising procedure, for range-independent variables there can never be a neuron tuned to multiple memories in the optimal solution.

Varying Mean Firing Rate: from Range to Statistical Independence

We see there are two extremes along an axis parametrised by t (with the two extremes presented in Section 10.A.4 and Section 10.A.5. If $t = 0$ (Section 10.A.5) then range-independence determines modularity - no matter how correlated the variables they will be encoded in different neurons. Conversely, if t is very large, all the firing rates will be high, and the nonnegativity constraint will have no effect. This is equivalent to the problem in Section 10.A.4, all that matters is the linear correlation between the variables. As you vary t you pass between the two regions, allowing you to explore regions where one or other effect dominates.

10.A.6 Solution III: Ansatz Solutions to the Full Problem

The full problem (Equation (10.28)) is non-convex and difficult to study[8]. In this section, rather than simplifying the loss, we instead simplify the potential tasks: we study a constrained set of tasks of direct relevance to the experiments we compare to El-Gaby et al., 2024; Y. Xie et al., 2022. Here we can propose ansatz solutions (guess solution classes), and show these accurately capture the behaviour of the (numerically) optimal representations. We begin by describing these constrained task settings, then we describe the shape of the ansatz optimal solution in the independent, and non-independent task settings.

Sequential Task Specifications

The tasks are fully specified by just two things: (1) the structure of the sequence and (2) the distribution of the observations. To match the sequential recall tasks used in the many experiments (e.g. El-Gaby et al., 2024; Y. Xie et al., 2022) we will study settings a task in which there is only a single action matrix, $\boldsymbol{W}$, that moves you forward through a sequence of observations. For convenience the sequence will be made cyclic[9], as we

[7] In fact, only a much looser property of extreme-point independence (across the set of datapoints when one variable is at its maximum or minimum value the other variables take both their maximum and minimum value) is necessary for this proof to work.

[8] In fact, there are even counter-examples that stop a similar proof to that in Section 10.A.5 from working. Precisely, there are arrangements of subspaces such that the recurrent loss is not minimsed by orthogonalising the subspaces while preserving within-subspace dot product structure (see code to demonstrate this).

[9] i.e. if the sequence is length L, then $\boldsymbol{W}^L = \mathbb{1}$, applying the forward matrix at the end of the sequence will return you to the start

discussed in Section 10.A.2. This matches the experiment in El-Gaby et al., 2024 but not Y. Xie et al., 2022, since the rodents in El-Gaby et al., 2024 cyclically repeat a task, while those in Y. Xie et al., 2022 only perform two full cycles. However, we contend that our analysis of cyclic tasks still applies to Y. Xie et al., 2022 since the non-cyclic theory states that memories will be forgotten after the final (second) cycle in Y. Xie et al., 2022, though it is reasonable to suggest the Monkeys still remember them (humans certainly can).

We assume a discrete set of N_O different possible observations and, for simplicity, we encode these as a demeaned one-hot N_O dimensional vector[10]. Since we have demeans the one-hot encodings, the vectors only spans $N_O - 1$ dimensions, breaking an assumptions we made earlier. We correct this by redefining $K = N_O - 1$.

To understand how the distribution of observations affects neural coding, we consider two different sampling choices in out tasks: sequence elements are either sampled independently, or without replacement. The entire task is therefore set by choosing the length of the sequence, the number of options per sequence element, and the sampling scheme. To understand the optimal neural code for this, we consider the following loss containing only the 'forward' matrix:

$$\mathcal{L} = \lambda_A \sum_{s,e} ||\boldsymbol{g}(s,e)||^2 + \lambda_R ||\boldsymbol{R}||_F^2 + \lambda_W ||\boldsymbol{W}||_F^2 \tag{10.81}$$

Orthogonal Encoding of Independent Memories

Inspired by the previous settings, we ansatz that in the independent case the memories are encoded in orthogonal subspaces. Since the encodings have to be nonnegative, orthogonal subspaces implies different neurons. Each subspace simply encodes the inputs in the positive orthant, with the minimal amount of firing. For one-hot encodings it is intuitive that the min-firing rate encoding that preserves both nonnegativity and the dot product structure of the data is a one-hot encoding. In order to make the mean-zero data-points positive we need to add a bias sufficient to overcome the negative elements of the encoding. For a positively-weighted (i.e. not multiplied by a negative number) one-hot code this is $\frac{1}{N_S}$, and we'll denote this number b_+. To maintain positivity, at least that much bias must be added; however, if the target firing rate is higher than b_+ then the representation will include extra bias. Therefore, we also permit each module to have a flexible bias that must be at least as large as b_+, but can otherwise be chosen to minimise the loss.

Now we construct an ansatz solution in which disjoint sets of neurons encode each of the memories:

$$\boldsymbol{g}(s,e) = \sum_{a=0}^{L-1} \mathbb{1}_a (\sigma_a \boldsymbol{o}_e(s+a) + b_a \mathbf{1}) \tag{10.82}$$

where we define $\mathbb{1}_a$ as a matrix that takes the D_O dimensional embedding and projects it to the aD_O to $(a+1)D_O$ dimensions of neural space. Hence the subspace dot product structure is:

$$\boldsymbol{D} = \begin{bmatrix} \sigma_0^2 \mathbb{1}_{D_O} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \sigma_1^2 \mathbb{1}_{D_O} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \sigma_{L-1}^2 \mathbb{1}_{D_O} \end{bmatrix} \tag{10.83}$$

Substituting this into the full loss (Equation (10.58)), it becomes:

$$\mathcal{L} = \lambda_A N_E (N_A \operatorname{Tr}[\Sigma] \sum_a \sigma_a^2 + D_O \sum_a (b_a - t)^2) + \lambda_R \frac{D_O}{\sigma_0^2} + \lambda_W D_O \sum_a \frac{\sigma_{a+1}^2}{\sigma_a^2} \tag{10.84}$$

where Σ is the correlation matrix of a single-memory observation:

$$\Sigma = N_A N_E \langle \boldsymbol{o}_e(s+a) \boldsymbol{o}_e^T(s+a) \rangle_{e,a} \tag{10.85}$$

We can then minimise this with respect to the parameters $\{\sigma_l, b_l\}_{l=0}^{L-1}$ to find the best representation, which is easily done numerically. Some high level intuition goes as follows though:

1. As λ_W goes to zero then only the readout subspace needs to have activity (due to the $\lambda_R \frac{D_O}{\sigma_0^2}$ term). The idea here is that the other subspaces only exist to fill the readout subspace at the appropriate time, the smaller the other subspace the more work has to be done by $\boldsymbol{W}$ to summon up their encodings. If you don't care about the size of $\boldsymbol{W}$ ($\lambda_W \to 0$) then these other subspaces can disappear.
2. As $\lambda_W \to \infty$ all the subspaces become the same size, since that makes life easiest for $\boldsymbol{W}$.
3. As λ_G becomes very large the representation shrinks to 0.
4. As λ_R becomes large the readout subspace becomes very large, so that it has an easy job decoding.

[10] Other encodings, including those derived from data, produce similar results. We discuss the effect of encoding scheme further in Section 10.A.7

Aligned Encodings of Memories Sampled without Replacement

The second case we consider is when observations for each sequence are sampled without replacement. In this case memories in different subspaces are negatively correlated with one another, and an efficient encoding of negatively correlated variables aligns them positively. This can be simply seen. Ignore positivity and imagine two variables, x_i and y_i, embedded in a neural encoding, $\boldsymbol{g}_i$, using encoding vectors $\boldsymbol{u}$ and $\boldsymbol{v}$:

$$\boldsymbol{g}_i = \boldsymbol{u}x_i + \boldsymbol{v}y_i \tag{10.86}$$

The average squared firing is:

$$\langle ||\boldsymbol{g}_i||^2 \rangle_i = ||\boldsymbol{u}||^2 \langle x_i^2 \rangle_i + ||\boldsymbol{v}||^2 \langle y_i^2 \rangle_i + 2\boldsymbol{u}^T\boldsymbol{v}\langle x_i y_i \rangle_i \tag{10.87}$$

So to reduce firing we can either reduce the size of the encodings ($||\boldsymbol{u}||^2$ and $||\boldsymbol{v}||^2$), or, more interestingly, if the variables are anticorrelated ($\langle x_i y_i \rangle_i < 0$), we can positively align their encodings ($\boldsymbol{u}^T\boldsymbol{v} > 0$). Following this logic, we ansatz a solution that has the same modular structure as before, but additionally includes M populations of neurons in which the encodings of the same stimuli across sequence elements are permitted to positively align:

$$\boldsymbol{g}(s,e) = \sum_{a=0}^{L-1}\sum_{a=0}^{L-1} \mathbb{1}_a(\sigma_a \boldsymbol{o}_e(s+a) + b_a\mathbf{1}) + \sum_{m=0}^{M} \mathbb{1}_{L-1+m}(\sum_{a=0}^{L-1} \mu_{m,a}\boldsymbol{o}_e(s+a) + \mu_{m,L}\mathbf{1})] \tag{10.88}$$

Crucially here, while the modular encodings are all positive, $\sigma_a > 0$, the mixed ones, $\mu_{m,l}$, can be negative, and in all cases the bias, $\mu_{m,L}$, is chosen so that the representation is at least nonnegative.

The subspace similarity matrix is equal to the sum of the subspace similarity within each module. The subspace similarity of a mixed module is:

$$\boldsymbol{D}_m = \begin{bmatrix} \mu_{m,0}^2\mathbb{1}_{D_S} & \mu_{m,0}\mu_{m,1}\mathbb{1}_{D_S} & \cdots & \mu_{m,0}\mu_{L-1}\mathbb{1}_{D_S} \\ \mu_{m,0}\mu_{m,1}\mathbb{1}_{D_S} & \mu_{m,1}^2\mathbb{1}_{D_S} & \cdots & \mu_{m,1}\mu_{L-1}\mathbb{1}_{D_S} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{m,0}\mu_{L-1}\mathbb{1}_{D_S} & \mu_{m,1}\mu_{L-1}\mathbb{1}_{D_S} & \cdots & \mu_{L-1}^2\mathbb{1}_{D_S} \end{bmatrix} \tag{10.89}$$

Combining these matrices across modules you can find the full subspace similarity, $\boldsymbol{D}$, and put that into the loss (Equation (10.58)). We can use equation solving packages to find the settings of $\{\sigma_l, b_l\}_0^L, \{\mu_{m,l}\}_{m,l=0}^{M,L}$ that minimise the loss.

10.A.7 Predictions for Modularising Trends

While there are intricacies, we can identify two broad currents that push representations closer or further from orthogonalising; and orthogonal encodings, when combined with nonnegativity constraints, can lead to single neurons tuned to single memories. These are some of our most testable predictions, so we outline the broad trends.

On the one hand, in Section 10.A.4 we examined how, without nonnegativity (or with a high enough target firing rate), the alignment of subspaces was largely driven by the correlation between memories. Positively correlated memories negatively align and vice versa. On the other, in Section 10.A.5 we studied how, when the target firing rate was 0, alignments are driven instead by range properties: range, and in fact even extreme-point independent, encodings are modularised. As the target firing rate varies, the influence of these two forces wax and wane. Further, since alignment is a signed quantity, these two forces can co-operate or compete depending on whether they push the subspaces to align in the same or opposite manner. For example, two memories might never take their minimal value at the same time, in which case they are usefully positively encoded in the same neuron, so that each memory might serve as a partial bias for the other. However, these range effects don't preclude those same memories from being positively correlated, an effect which would, on its own, encourage negative alignment, inducing a tussle between the two effects. We can use these broad trends to understand various, otherwise puzzling, patterns of modularisation in our optimal representations.

For a simple example, consider the effect of increasing the number of stimuli in a simple sequential memory task, as we studied in section Section 10.A.6. As before, if the sequence elements are independent then their encodings will be modularised. On the other hand, sampling without replacement makes the memories anticorrelated. Further, if encoded using one-hot then paired dimensions (e.g. both first dimensions) of two memories will never concurrently attain their maximum encouraging the formation of a representation in which both memories are negatively encoded in the same neurons, positively aligning their representations. As such, both forces are encouraging positive alignment of the memories. Increasing the number of stimuli, while leaving the range properties the same, reduce the correlations between stimuli - in the limit case with infinite stimuli sampling without replacement has basically no effect. There is a threshold number of stimuli, below this number the subspaces align, above they modularise, Figure 10.9a.

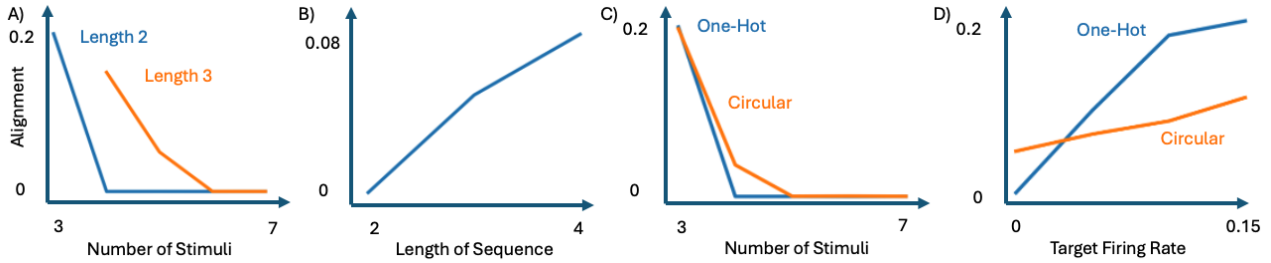


Figure 10.9: Broad trends in the alignment of subspaces in simple sequential tasks, Section 10.A.6. In all cases $\lambda_A = 0.1$, $\lambda_W = \lambda_R = 1$, the stimuli are sampled without replacement, and we report the alignment between the first and second subspace. A) For one-hot encoded sequences of length 2 and 3 with target firing rate equal to 0 ($t = 0$), increasing the number of stimuli decreases the alignment, until a critical number beyond which all sequences are modularised. B) For one-hot coded memories with five stimuli and $t = 0$, increasing the sequence length moves you from modularising (length 2) to increasingly aligning. C) Different encodings modularise at different points, plotted are one-hot and circular alignment for length 2 sequences with $t = 0$ as a function of the numbers of stimuli. D) Increasing the target firing rate increases alignment across two encodings of length 2, four stim sequences.

A similar example comes from increasing the length of the sequence. For a fixed number of stimuli and encoding, increasing the sequence length of a sampled without replacement sequence increases the correlation, encouraging alignment, Figure 10.9b. For example, this might manifest in a higher critical number of stimuli for higher sequences lengths.

The encoding scheme is also a vital consideration. For example, contrast the effect of encoding memories using either a one-hot or a 2D-angular scheme. For the same sequence structure, angular encodings have less range missing making them less likely to modularise. However, their range is symmetric around their mean (meaning that the distance between the mean and the max and min along a particular encoding direction are equal, unlike in a one-hot case), which makes the missing portion of range more potent. As such, we can find cases where the one-hot code is modularised by the circular encoding is aligned, Figure 10.9c.

Finally, we've discussed how increasing the target firing rate increases the importance of linear correlation, eventually ignoring range properties. In particular, since linearly correlated data, such as sequences sampled without replacement, should always align in the linear model and it is only the nonnegativity that sometimes impedes this, as you increase the target firing rate, getting closer to linearity, the alignment increases, Figure 10.9d.

All of these effects point to a complex landscape of modularising and aligning. For 1-dimensional memories with zero target firing rate, we can derive, in a small extension of our previous work William Dorrell, K. Hsu, et al., 2025, necessary and sufficient conditions on the range and correlation of the memories to modularise. This is a nice theoretical result, precisely outlining the line that divides modularising from aligning representations, though it does not give us a quantitative handle on the level of induced alignment. Deriving similar results for multidimensional memories is, however, challenging. Despite that, these rough trends can be intuitively understood using the conceptual framework presented, and make directly testable predictions for experiments.

10.B Numerical Simulations

In this appendix we discuss the numerical results presented. We begin by outlining our optimisation schemes, then the different tasks. Finally, we describe the analyses used to extract representational properties and produce the plots.

10.B.1 Optimisation Schemes

Our simulations broadly fall into three categories. The first aligns more closely with the mathematical framing. We use the derived subspace structure to write all representations as an affine function of a set of lagged observations. We then optimise over this choice of affine function. The second is similar, except we skip the affine map parameterisation, and directly optimise over the neural representation matrix itself. The third set is exactly a gated-linear RNN: a linear RNN with action-dependent recurrent weight matrices. In this case we optimise over the weights and biases, as in a traditional RNN setup.

Optimising over Affine Readin Matrix

We saw in section 10.A.2 that, after experiencing every observation, the representation can be written as a sum of a set of subspaces, each encoding the memory at a particular offset, and a bias. In this optimisation scheme we make use of this. We don't model the representation while the memories are being inputted. Rather, we only model the representations once they are 'full' - once every observation has been seen once. This is a natural fit for tasks like those studied by El-Gaby et al. (2024) and Basu et al. (2021) in which the animal repeats one sequence for a long time before changing. Then we should expect the input activity to have minimal effect. This is less natural for more trial based tasks, as studied by Y. Xie et al. (2022) and Panichello and Buschman (2021), but we use it anyway. We consider the more natural parametersiation of these tasks, a standard RNN approach, in the next section, which we use for some of our results when necessary.

The neural representation is a function of two things (1) the pairing between state and stimuli in the current task and (2) the current position. Following our discussion above, we create these representations using an affine map and a set of lagged inputs. These lagged inputs describe the stimuli at a particular offset from the current position, in the current task.

Our full loss that is used to train these representations is as follows:

$$\mathcal{L} = \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{compute}}\mathcal{L}_{\text{compute}} + \lambda_{\text{pos}}\mathcal{L}_{\text{pos}} + \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{R}}\mathcal{L}_{\text{R}} + \lambda_{\text{W}}\mathcal{L}_{\text{W}} \tag{10.90}$$

The first three implement the constraints (functional, actionable, nonnegativity (if enforced, else $\lambda_{\text{pos}} = 0$), hence their coefficients are large. The remaining three form the objective. We now describe each in turn.

Readout Loss and Functional Constraint We consider the representation as implicitly defining a readout map. Given a representation matrix, $\boldsymbol{G} \in \mathbb{R}^{N \times N_S N_E}$, and a set of observations, $\boldsymbol{O} \in \mathbb{R}^{K N_A \times N_S N_E}$ we construct the minimal L2 norm readout matrix. Since the readout includes a bias that is not penalised, it makes use of it to perform all parts of the readout mapping that involve predicting a constant output. We can therefore think of the readout matrix as acting not on the representation and observations directly, but on their demeaned partners, $\bar{\boldsymbol{G}} = \boldsymbol{G} - \boldsymbol{G}\boldsymbol{1}\boldsymbol{1}^T$ and similarly $\bar{\boldsymbol{O}}$. Then the minimal L2 readout matrix which predicts as much of the data as possible is given by the pseudoinverse:

$$\boldsymbol{R} = \bar{\boldsymbol{O}}\bar{\boldsymbol{G}}^{\dagger} \tag{10.91}$$

Then we can measure the reconstruction error and readout weight norm:

$$\mathcal{L}_{\text{recon}} = ||\boldsymbol{R}\bar{\boldsymbol{G}} - \bar{\boldsymbol{O}}||_F^2 \qquad \mathcal{L}_{\text{R}} = ||\boldsymbol{R}||_F^2 \tag{10.92}$$

Recurrent Loss and Computing Constraint The procedure for the recurrent weight is very similar, except rather than mapping representation to output, we map representation to permuted representation. Hence we construct the following min-norm matrix:

$$\boldsymbol{W}_a = \bar{\boldsymbol{G}}\boldsymbol{P}_a\bar{\boldsymbol{G}}^{\dagger} \tag{10.93}$$

where $\boldsymbol{P}_a$ implements the appropriate permutation. For example, if the task had a set of three states and the action looped you through them, it would be:

$$\boldsymbol{P}_a = \begin{bmatrix} 0 & 1 & 0 & & & & \dots \\ 0 & 0 & 1 & & \mathbf{0} & & \dots \\ 1 & 0 & 0 & & & & \dots \\ & & & 0 & 1 & 0 & \dots \\ & \mathbf{0} & & 0 & 0 & 1 & \dots \\ & & & 1 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{10.94}$$

Then the losses are:

$$\mathcal{L}_{\text{compute}} = \sum_A ||\boldsymbol{W}_a \bar{\boldsymbol{G}} - \bar{\boldsymbol{G}} \boldsymbol{P}_a||_F^2 \qquad \mathcal{L}_{\text{R}} = \sum_a ||\boldsymbol{W}_a||_F^2 \tag{10.95}$$

Nonnegativity and Activity The final two are simple:

$$\mathcal{L}_{\text{pos}} = -\sum_{nt} \text{ReLU}(-G_{nt}) \qquad \mathcal{L}_{\text{act}} = ||\boldsymbol{G}||_F^2 \tag{10.96}$$

Optimising over Representation

This approach is identical to the previous one except, rather than creating $\boldsymbol{G}$ using the affine map from a set of lagged inputs, we directly optimise over $\boldsymbol{G}$. Otherwise the optimisations are identical. This drops the constraint that the representation is an affine function of the input, but preserves all others. To be coherent, it would be better to run these simulations in one of the other two frameworks.

Optimising a Gated Linear RNN

The final alternative is more similar to a neural network. We specify a set of inputs, a set of outputs, and a set of actions which determine which recurrent weight matrix to use. We then construct the representation by feeding inputs and implementing the gated-linear RNN equation:

$$g_{t+1} = \boldsymbol{W}_{a_{t-1}} \boldsymbol{g}_{t-1} + \boldsymbol{W}_0 \boldsymbol{i}_t + \boldsymbol{b} \tag{10.97}$$

Then we follow the same procedure above, except using the explicitly parameterised $\boldsymbol{W}_a$ matrices, and ignoring the computing constraint, since the RNN formulation enforces it architecturally.

10.B.2 Task Details

We now outline each of the tasks. Specifying a task involves specifying the sets of memories, and the actions. Additionally, in the RNN setting, we have to specify the inputting scheme.

Affine Parameterisations

In all cases we use a single action that periodically loops through the sequence.

El-Gaby: 4 observations sampled from a potential 9 without replacement and encoded one-hot.

Mushiake: 3 observations sampled from a potential 4, encoded one-hot. We used only the 24 sequences used in the experiment.

Xie: 3 observations sampled without replacement from 6, encoded one-hot.

Nakajima: 2 observations, sampled from 4 possible 2-hot codes made from a combination of 2 2-dimensional one-hot codes. We either sampled the two observations independently, producing 16 sequences, or removed one pairing (e.g. left arm never supinated) producing 9 sequences.

Harmonic Explorations: We concurrently ran 2 sequence tasks. In the anharmonic case one was an ABC loop, the other an EF loop, for a total sequence length of 6. Each observation was either 0 or 1, producing a 2 dimensional output - one dimension for each sequence. Correlations between tasks were introduced by only sampling sequences in which A was different to E. In the harmonic but independent case we switched to ABCD and EF, resulting in a total period of 4, and sampled each observation independently. Finally, we again introduced correlations in this case by only sampling sequences in which A was different to E.

Phase: Similarly, we had two concurrent length 3 sequences, one ABC, the other 001, producing a 2D prediction. In the independent case we sampled all combinations of observations. In the dependent case we ensured A was never 1, hence observation 1 never occurs when you are rewarded.

Overlapping Subspace: We use two sequences length 3 sequences. In the deterministic case all 6 stimuli are unique. In the slightly overlapping case one stimuli overlaps between the two.

Neural Parameterisation

In this framework there is no input, by setting the output and optimising the representation, it implictly learns to encode the input, else it can't solve the task.

Panichello & Buschman: We sample 2 observations from a set of 4 either with or without replacement. We then setup a task with three timesteps: a delay period, and two potential readout futures, up or down. To create this, we setup a representation matrix of size (number of neurons)$\times$(number of tasks$\times$3). The first (number of tasks) representations correspond to the delay period, the next to the recall of the top memory, the last to the bottom memory. There are two action matrices, a top and a bottom, which are constructed to move the delay period activities to their corresponding top or bottom readout. Finally, a readout matrix decodes the labels from the readout representations, either the identity of the bottom cue from the last (number of task) representations, or the top for the middle (number of task) representations.

Stroud: This task has two one-hot encoded stimuli and a variable delay time, between 2 steps and 4, after which there is a 2 step readout time period. There is one forward action that operates at all timepoints except when the 'go' cue arrives. The three tasks differ in their readout requirements. In the first, 'cue-delay', the readout has to predict the correct memory from the beginning of the delay period until the end of the task. In the second, 'after-go-time', the readout only has to predict the stimuli in the readout period. In the third, 'just-in-time', the representation must correctly predict from the end of the shortest delay period until the end of the task.

RNN Parameterisations

Motor - Zimnik: We trained a ReLU RNN to reach. Each reach is defined as a one-hot code. The one-hot part of the network's input tells it which reach it will have to perform next, but not when. A final dimension of the input encodes the go signal saying 'on the next time step do the cued reach!'. The network is then trained to output the cued-one-hot code for a couple of timesteps after the go cue. The network is either trained on a variety of single-reaches, or two reaches cued one after the other.

Xie - Different Algorithms: We train gated linear RNNs on a task with up to 6 timesteps: up to 2 inputs, either 0 or 1 delay steps, and up to 2 outputs. Each sequence is made of 1 or 2 stimuli, out of a possible 4, sampled from a circle. We sample length two sequences without replacement, and in sequence with only 1 item, it could arrive in either the first or second timestep, so its a bit more like the other element was empty rather than a different length sequence.

The actions are chosen according to two different schemes. In both cases, in order to construct an input gating network, two distinct recurrent matrices are used for the two input timesteps. Then there is a delay action. Finally there are either two distinct output actions applied after a go cue, to generate an output gating mechanism, or a single output action, to generate a conveyor belt.

10.B.3 Analysis

Here we outline how we extracted various properties of the representation.

Simple: tuning curves, and dot-products Some of the plots are very simple. In particular, fig. 10.4J and M just contain tuning curves: the average activity of some simulated neurons for different combinations of movements at different lags. A similarly easy plot is the bottom row of fig. 10.5F, which is made by using activity from the longest delay period and taking the dot-product between the activity at different time points.

Proportion of Variance To calculate the proportion of variance explained by each task, we first calculate the proportion of variance explained by each lagged stimuli. To do this, we calculate the variance in each neuron's activity conditioned on each value of the lagged stimuli, which in this case was either 0 or 1. We then calculate what proportion of the neuron's total variance was removed in this condition process, by taking the total variance, subtracting the conditioned variance, and dividing by the total variance. Finally, to turn it into a task-specific metric we sum up all contributions from lagged variables belonging to the same task.

Subspace properties - Affine parameterisation This setting has no special input structure we could to track the emergence of subspaces, hence it is as difficult as extracting subspaces from real neural data. We therefore use the same techniques, and defer their explantion until section 10.C.

Subspace properties - Different Algorithm Gated Linear Networks To extract subspace properties from the gated-linear RNN network we simply pass a single observation through the network and observe its passage after different combinations of actions. We then do this for all single observations, and, for a given action sequence, we construct the subspace using the top 2 PCs of these activities. We then project each stimulus encoding into this 2D space. To measure the size of the space we simply take the average norm of these vectors. To create the information flow plots we use the same subspaces, but project activity onto it from all 2 sequence trials.

Subspace Alignment - Motor Networks To match the subspace alignment metrics used on previously reported motor data (Elsayed et al., 2016), we used a different alignment metric for the motor example, fig. 10.4G. We gather the activity in the network during single-reach tasks and break it into preparatory - activity after the cue has been presented but before the network outputs anything - and potent - activity during the reach - activity. We then perform PCA on these two sets of activity. We build subspaces made from a subset of these principal components: if the neural activity projected along a principal component captures more than 0.1% of the total variance of that neural activity (preparatory or potent) we include it. We then measure what proportion of the variance in one type of neural activity (preparatory or potent) is captured after projecting to the other subspace.

10.C Extracting (Correlated) Memory Subspaces from Data

We analysed neural data from Panichello and Buschman (2021) and Y. Xie et al. (2022). We are incredibly grateful to both groups for sharing their data with us.

In both cases we extracted the sizes and alignments of the subspaces encoding each memory during the delay period of the task, and we performed similar analysis on some of our theoretical representations. In this appendix we consider three methods for extracting vectors that describe the subspaces, and we construct a signed measure of alignment.

The basic problem goes as follows. We assume we have some data linearly generated by a set of variables:

$$\boldsymbol{g}_t = \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{V}\boldsymbol{y}_t + \boldsymbol{b} \tag{10.98}$$

How do we estimate $\boldsymbol{U}$ and $\boldsymbol{V}$ from a dataset of $\{\boldsymbol{g}_t, \boldsymbol{x}_t, \boldsymbol{y}_t\}$. So far, so easy. But crucially, how do we do it in a way that our estimate of the alignment of the two subspaces, $\boldsymbol{U}^T\boldsymbol{V}$, is unbiased?

The first method, averaging, is the simplest, but only works for independently sampled memories. The second, regularised regression, was used by Y. Xie et al. (2022) and works, but is sometimes biased (not too badly). The third we invented for specifically this purpose, and works by doing regression on differences of representations.

This section begins by outlining the preprocessing steps we performed to get the measured representations to the same state as our theoretical ones. Then, we discuss method for estimating the alignment between a set of subspaces, and problems with them, leading to our own metric. Finally, we try and estimate subspaces from data. We begin with the averaging method, how we used it to analyse the Panichello and Buschman (2021) data, and how it fails on correlated memories, producing a biased estimate. Then we describe the other two methods, their results on synthetic data, and how we used them to analyse the data of Y. Xie et al. (2022).

10.C.1 Preprocessing

Panichello & Buschman: We worked with neural data labelled as 'frontal'. We binned the stimuli (really a continuous angle) into 4 bins and only analysed trials on which the monkey was within half a radian of the target. We extracted the firing rate of the neurons during the last half second of the delay period.

Xie et al.: We used all neurons from correct trials, and extracted the last second of $\frac{\Delta F}{F}$.

10.C.2 Alignment Metric

Existing alignment methods use angles. Panichello and Buschman (2021) create two 2-dimensional subspaces in the top 3 PCs then measure the angle between their normals, while Y. Xie et al. (2022) instead report the first principal angle between the 2D subspaces. Both have their shortcomings. The first method relies on projecting to 3D rather than dealing with the high-dimensional space directly. The second, by relying on only the first principal angle, introduces a bias. The principal angles are ranked, the first is smaller than the second etc. This means, even if a 2D subspace is oriented at a constant angle to another, if there is a small amount of noise it will tend to push the first principal angle smaller and the second larger, as in the synthetic data example in fig. 10.10. This particular effect can be removed by averaging the principal angles.

However, a second problem remains. Angles are capped at 90°. Let's say your the true average principal angle is close to orthogonal. You try to estimate this from some noisy data. Perhaps some principal angles go down and some of them up. Due to the range capping the ones that went up could only go up to 90°, and if they are noisy enough they might go from being slightly aligned in one direction to slightly aligned in the opposite direction. Since the values we are calculating don't take account of the directionality of the alignment, this means that when you add noise the estimate of the principal angles slips away from orthogonal, fig 10.11. This suggests that any value we estimate from data is likely to be an overestimate of any close to orthogonal alignment between subspaces.

This problem occurs because of the bounding of the angle. A better estimate would be a signed estimate of alignment that was 0 if the subspaces were orthogonal, positive if the encoding of the same stimulus in different subspaces align, and negative if they don't. We therefore calculate the average cosine similarity between pairs of the same stimulus across the two subspaces, which avoids all these problems.

10.C.3 Estimating Subspaces by Averaging

We begin with our first way of estimating the subspaces. If $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are independent then life is easy. Averaging all $\boldsymbol{g}_t$ for a given $\boldsymbol{x}$ produces an estimate of a shifted projection of $\boldsymbol{U}$, since the average of $\boldsymbol{y}_t$ conditioned on $\boldsymbol{x}_t$

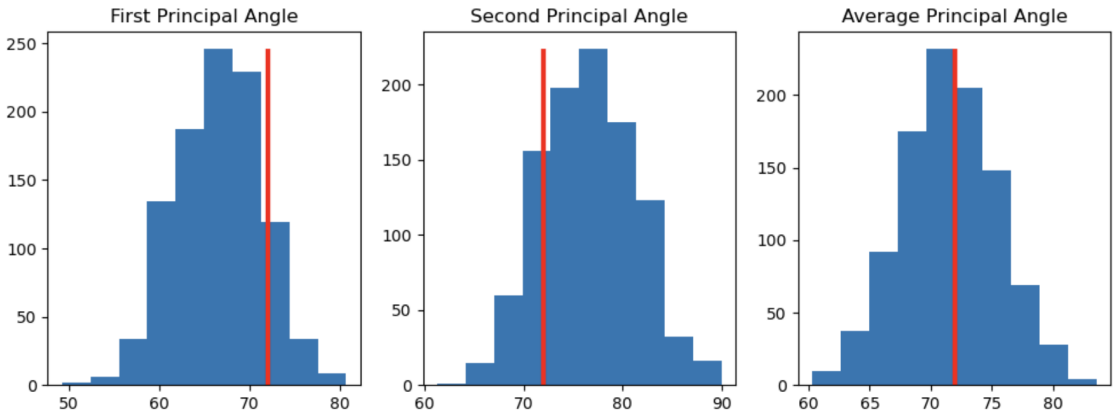


Figure 10.10: We create some fake data of two 2D subspaces that are aligned at 72°, i.e. both of the two principal angles are 72°, the red line. I then add a small amount of noise 1000 times and calculate the principal angles between the noisy vectors (all the vectors are length one, and I add a zero-mean gaussian noise matrix with variance 0.1). As you can see, the estimate of the first principal angle is biased down, the second up, but the average appears unbiased.

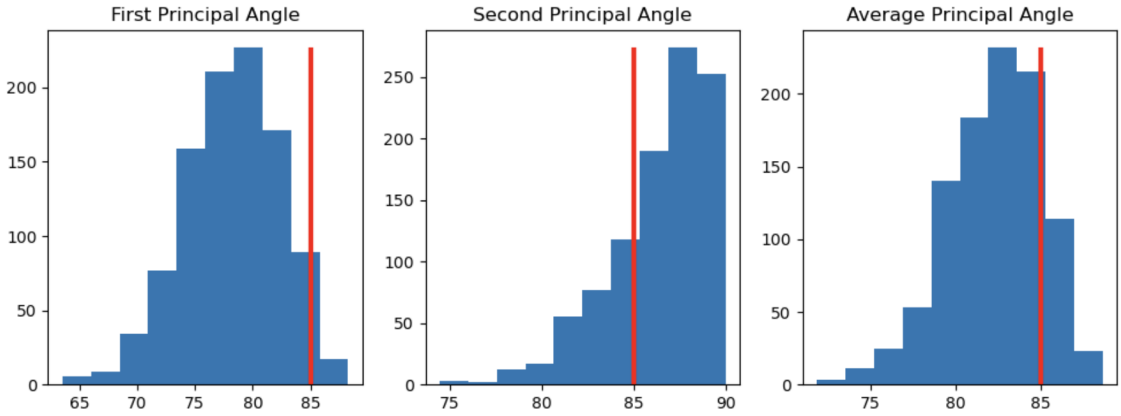


Figure 10.11: Same setting as figure 10.10, except now the true principal angle, the red line, is 85°. We can see that all estimators, including the average principal angle are shifted downwards.

is the same as the unconditioned average of $\boldsymbol{y}_t$:

$$\langle \boldsymbol{g}_t \rangle_{t:\boldsymbol{x}_t=\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{x} + \langle \boldsymbol{V}\boldsymbol{y}_t \rangle_{t:\boldsymbol{x}_t=\boldsymbol{x}} + \boldsymbol{b} = \boldsymbol{U}\boldsymbol{x} + \boldsymbol{V}\langle \boldsymbol{y}_t \rangle_t + \boldsymbol{b} \tag{10.99}$$

Doing this for all $\boldsymbol{x}$ creates a set of vectors, then performing PCA we create a 2D subspace that encodes these memories. To get a size estimate we project each average $\boldsymbol{x}_t$ encoding onto this subspace and measure the average size of the projection, and to we calculate alignments using the previously measured metric.

Failing for Correlated Memories Imagine now that you have correlated the memories (which wasn't actually a problem for Panichello and Buschman (2021)). For example, imagine two perfectly orthogonal subspaces encoding a set of 4 direction, N, S, E, W, fig. 10.12. You now take the average conditioned on the first stimulus being north. Clearly the conditioned average activity in the first subspace points in a consistent direction, so far so good. However, because of the without replacement sampling, the north stimuli is never in the second subspace during these trials. Hence, in the second subspace the conditioned average points south.

Now that you have your biased average 'stimulus 1 = north' activity, you take the dot product with 'stimulus 2 = north' average. Despite the fact the subspaces are orthogonal you find a negative alignment! The anticorrelations in the variables manifests as a predicted negative correlated between the subspaces.

10.C.4 Regularised Regression

Y. Xie et al. (2022) had to come up with a clever method to circumvent this problem: they performed regularised regression. Assume each memory is a linear combination of different subspace encodings, writing the neural

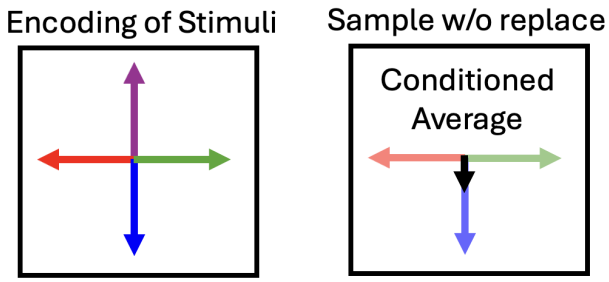


Figure 10.12: Imagine each subspace contains data pointing in the 4 cardinal directions. Between the two subspaces these directions are sampled without replacement. This means the average activity in the second subspace when the first subspace is pointing north will actually point south.

activity for a sequence of two angles θ_1 and θ_2 as follows:

$$\boldsymbol{g}\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right) = \boldsymbol{g}_1(\theta_1) + \boldsymbol{g}_2(\theta_2) + \boldsymbol{c} \tag{10.100}$$

Stack each of the encoding vectors into a matrix, the coefficients of the regressors:

$$\beta = \begin{bmatrix}\boldsymbol{g}_1(\theta_1) & \dots & \boldsymbol{g}_1(\theta_Q) & \boldsymbol{g}_2(\theta_1) & \dots & \boldsymbol{g}_2(\theta_Q)\end{bmatrix} \tag{10.101}$$

Then for each sequence you create a two-hot (in this case, because there's two sequence elements) vector that pulls out those vectors, let's call it $\boldsymbol{t}_{i_1,i_2}$. Minimise the following LASSO regression loss:

$$\mathcal{L}_{\text{reg}} = \sum_{\text{sequences}} \left| \boldsymbol{g}\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right) - \beta \boldsymbol{t}_{\theta_1,\theta_2} \right|^2 + \lambda|\beta|_1 \tag{10.102}$$

The λ is chosen by splitting the data in half and finding the value which causes the error on the second half of the data to be minimised. This method is much better: on noiseless data it estimates the quantity correctly, fig. 10.13.

Unfortunately, on noisy data it is biased, fig. 10.13. This can be more easily understood with L2 regularised regression, and the same insights appear to carry over to L1.

Ordinary Least Squares Estimator The regularised L2 regression weights for mapping from the stacked matrix of regressors, $\boldsymbol{T} \in \mathbb{R}^{R\times T}$, where R is the number of regressors and T the number of datapoints, to stacked neural activity $\boldsymbol{G} \in \mathbb{R}^{N\times T}$ is:

$$\beta = \boldsymbol{G}\boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{T}^T + \lambda\mathbb{1})^{-1} \tag{10.103}$$

We are interested in the dot-product of the regressors. These are second order statistics, so they are governed by the covariance of the estimator:

$$\mathbb{V}[\beta] = (\boldsymbol{T}\boldsymbol{T}^T + \lambda\mathbb{1})^{-1}\boldsymbol{T}\boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{T}^T + \lambda\mathbb{1})^{-1} \tag{10.104}$$

If λ is small (low noise), this is:

$$\mathbb{V}[\beta] = (\boldsymbol{T}\boldsymbol{T}^T)^{-1} \tag{10.105}$$

This means that, even if the subspaces are orthogonal, if the regressors are positively correlated $(\boldsymbol{T}\boldsymbol{T}^T)_{ij} > 0$ the estimate will tend to be negative.

Alternatively, if λ is large (high noise):

$$\mathbb{V}[\beta] = (\boldsymbol{T}\boldsymbol{T}^T) \tag{10.106}$$

So now the alignment estimate follows the correlations of the regressors!

Therefore, varying λ will push your estimate from positively biased to negatively biased. Somewhere in the middle is a perfect λ value, and held-out data does quite a good job of finding it for low noise levels. However at high noise levels it fails, fig. 10.13. I'm not sure why, but empirically it tends to choose an overly large λ, leading to estimates that align with the correlations in the data.

10.C.5 Difference Method

We present an alternate method for estimating subspaces. We simply take the difference of encodings that differ only in the subspace we are interested in:

$$\boldsymbol{g}\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right) - \boldsymbol{g}\left(\begin{bmatrix}\theta_1'\\ \theta_2\end{bmatrix}\right) = \boldsymbol{g}_1(\theta_1) - \boldsymbol{g}_1(\theta_1') \tag{10.107}$$

For all pairs of sequences that differ in only one element we take the difference, producing a vector that lives in one subspace (+ noise). We use the average length of these vectors as our size estimate for each subspace, and we calculate the average dot product between the same difference in two subspaces, e.g.:

$$(\boldsymbol{g}_1(\theta_1) - \boldsymbol{g}_1(\theta_1'))^T(\boldsymbol{g}_2(\theta_1) - \boldsymbol{g}_2(\theta_1')) \tag{10.108}$$

On noisy synthetic data this is also biased, fig. 10.13, but in ways that are conducive to our aims. As the noise increases all the alignments collapse to zero, i.e. the alignments never switch from positive to negative. This is perfect for arbitrating hypotheses like 'these subspaces are positively aligned'. If our estimator says it is, then that is *despite* the noise, rather than aided by it, as can be the case for the regression estimator.

Conversely, neither estimator mistakes a smaller subspace for a bigger one, and it turns out that the regression estimator is much better at estimating these quantities. We therefore use the regression estimator for sizes and the difference estimator for angles.

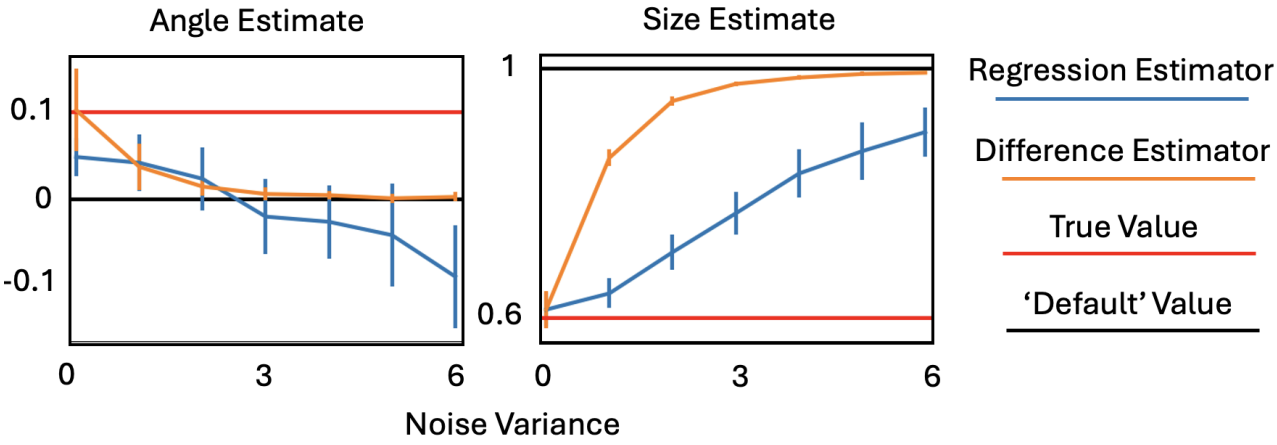


Figure 10.13: We create data from two subspaces aligned at 0.1 with size ration 0.6 and add Gaussian noise. We use the two estimators to find the size ratio and angle across 100 noise draws and plot the mean and standard deviation across different noise levels. Both are good estimates at low noise levels. At high noise levels the difference method returns to 'default' estimates, while the angle estimated by the regression estimator heads negative. This motivates using the difference method for alignments. However, the regression estimator's estimate of size is much better than the difference method's for longer, motivating its use for size ratios.

10.D Overlapping Sequence Coding

In the limit of deterministic sequences the subspace code collapses, and we are left with a pure code for observation which, due to the structure of the task, is synonymous for the conjunction of position within task and task identity. We provide a simple in fig. 10.14, a representation trained to predict two sequences of length three with distinct one-hot stimuli, producing a sequence-specific encoding of position, or in other words, a place cell code for stimuli.

However, if we add a small amount of indeterminancy, a single stimulus that appears in both sequences, then the neural code must distinguish these two settings in order to predict the rest of the sequence. It does this with a mixture of neurons, many of which encode the repeated stimulus in a sequence specific way, as highlighted, fig. 10.14.

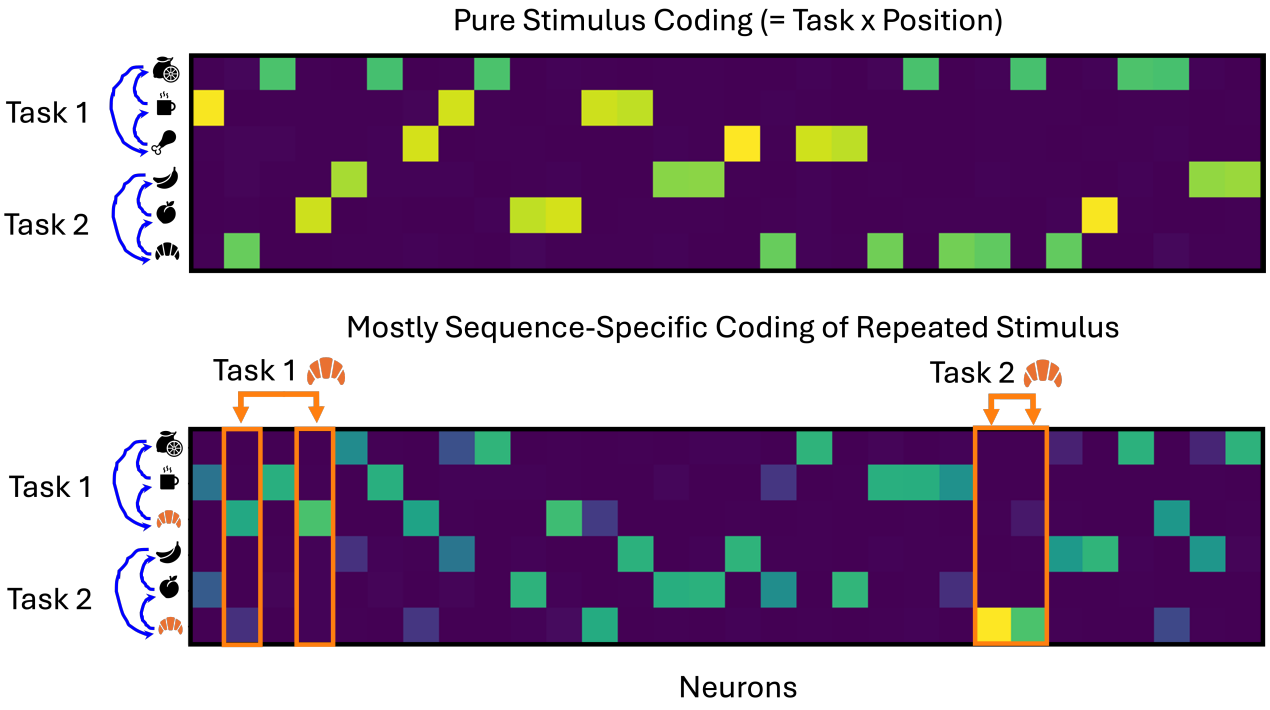


Figure 10.14: We train a representation on a task with two length 3 periodic sequences. Each memory is distinct, so there is no compositional structure. The representation, naturally, learns a place cell code for object identity. Including a repeated sequence element leads the optimal representation to include neurons that encode the repeated stimuli differently in the two contexts.

Part IV

Connectivity & Generalisation in Cerebellar-like Networks

Remarkably, each of us previously wasn't. At some point in the last century we each came into existence, uselessly babbling, unable to coordinate our limbs or marshal our thoughts. Now look at us: capable of walking, talking, directed thought; all the many abilities and choices that, while perhaps not being best used, are at least being used to read this thesis. This transformation is a marvel. Curiosity regarding this marvel has directed much work in neuroscience, psychology and computer science towards understanding the learning question: how do neural systems like us effectively learn from experience?

Now, there are many types of learning that are often hard to disentangle from one another. Some occur as animals develop, iteratively bootstrapping their circuits into operation using nothing more than force of will and millions of years of evolutionary know-how. Others occur over the animals lifetime, storing the memories that make up our identities. This thesis does not concern these types of learning.

Another, much more circumscribed form of learning involves adaptive algorithms. Take the example from the previous section: remembering and repeating a sequence of four locations (El-Gaby et al., 2024). In some sense this is working memory; but in another it is one fixed adaptive algorithm - 'remember the four rewarded locations and repeat them until further notice'. Having chosen the fixed algorithm we were then able to reason about its optimal neural implementation - the north star of this thesis.

However, the previous section also noted an equivalent formulation of the algorithm. Rather than solving the whole problem in activity, we could use weight changes to encode the rewarded locations - matching more traditional formulations of things like episodic memory. Can we extend our normative theorising to settings where plasticity does occur online during the algorithm's performance?

This chapter presents two attempts to develop or describe tools to normatively reason about circuits performing ongoing in-weights learning. In particular, we target supervised learning in cerebellar-like networks, which comprise some of the most cleanly understood circuitry in neuroscience. Our goals are substantially less comprehensive than those in previous sections. Ideally, analogously to the route taken in the rest of the thesis, we would frame a broad problem, like learning from a limited set of data an appropriate generalisation to unseen data, and derive a solution that matched biology. Here, instead, we answer a much narrower set of questions. We develop tools to link design choices in cerebellar-like networks to the learning abilities they endow in the network, allowing us to give them normative justifications. A solution to the grander version of the question will have to wait for someone else's thesis.

The two sections describe the two approaches taken. The first is a novel meta-learning framework which we use to learn functions that networks are good at learning. The second is a pedagogical application of an existing link between cerebellar-like networks and kernel regression, which allows us to analytically describe the generalisation properties of the network. Both techniques allow the linking of design choices to the learning ability of the circuit.

Chapter 11

Meta-Learning the Inductive Bias of Simple Neural Circuits

Training data is always finite, making it unclear how to generalise to unseen situations. But, animals do generalise, wielding Occam's razor to select a parsimonious explanation of their observations. How they do this is called their inductive bias, and it is implicitly built into the operation of animals' neural circuits. This relationship between an observed circuit and its inductive bias is a useful explanatory window for neuroscience, allowing design choices to be understood normatively. However, it is generally very difficult to map circuit structure to inductive bias. Here, we present a neural network tool to bridge this gap. The tool meta-learns the inductive bias by learning functions that a neural circuit finds easy to generalise, since easy-to-generalise functions are exactly those the circuit chooses to explain incomplete data. In systems with analytically known inductive bias, i.e. linear and kernel regression, our tool recovers it. Generally, we show it can flexibly extract inductive biases from supervised learners, including spiking neural networks, and show how it could be applied to real animals. Finally, we use our tool to interpret recent connectomic data illustrating its intended use: understanding the role of circuit features through the resulting inductive bias.

11.1 Introduction

Generalising to unseen data is a fundamental problem for animals and machines: you receive a set of noisy training data, say an assignment of valence to the activity of a sensory neuron, and must fill the gaps to predict valence from activity, Fig. 1A. This is hard since, without prior assumptions, it is completely underconstrained. Many explanations or hypotheses perfectly fit any dataset (D. Hume, 1748), but different choices lead to wildly different outcomes. Further, the training data is likely noisy; how you sift signal from noise can heavily influence generalisation, Fig. 1B.

Generalising requires prior assumptions about likely explanations of the data. For example, prior belief that small changes in activity lead to correspondingly small changes in valence would bias you towards smoother explanations, breaking the tie between options 1 and 2 in Fig. 1A. It is a learner's inductive bias that chooses certain, otherwise similarly well-fitting, explanations over others. Beyond this, the inductive bias is not rigorously defined. For the purposes of this paper we operationalise the inductive bias via the generalisation error: learners are inductively biased towards functions they generalise with low error.

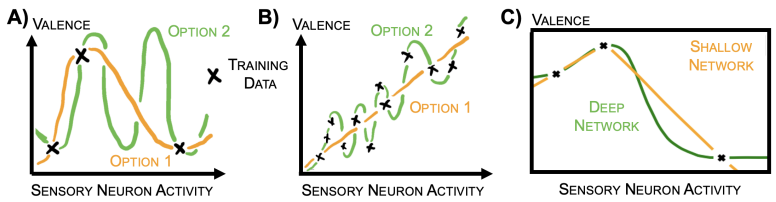


Figure 11.1: **Generalisation Requires Prior Assumptions. A:** The same dataset is perfectly fit by many functions. **B:** Different assumptions about signal quality lead to different fittings. **C:** Training a 2 (shallow) or 8 (deep) layer ReLU network on the same dataset leads to different generalisations.

The inductive bias of a learning algorithm, such as a neural network, can be a powerful route to understanding in both Machine Learning and Neuroscience. Classically, the success of convolutional neural networks can be

attributed to their explicit inductive bias towards translation-invariant classifications (LeCun et al., 1998), and these ideas have since been successfully extended to networks with a range of structural biases (Bronstein et al., 2021). Further, many network features have been linked to implicit regularisation of the network, such as the stochasticity of SGD (Mandt, Hoffman, and Blei, 2017), parameter initialisation (Glorot and Bengio, 2010), early stopping (Hardt, Recht, and Singer, 2016), or low rank biases of gradient descent (Gunasekar et al., 2017).

In neuroscience, the inductive bias has been used to assign normative roles to representational or structural choices via their effect on generalisation. For example, the non-linearity in neural network models of the cerebellum has been shown to have a strong effect on the network's ability to generalise functions with different frequency content (M. Xie et al., 2022). Experimentally, these network properties vary across the cerebellum, hence this work suggests that each part of the cerebellum may be tuned to tasks with particular smoothness properties. This is exemplary of a spate of recent papers applying similar techniques to visual representations (Bordelon, Canatar, and Pehlevan, 2020; Pandey et al., 2021), mechanosensory representations (Pandey et al., 2021), and olfaction (K. D. Harris, 2019).

Despite the potential of using inductive bias to understand neural circuits, the approach is limited, since mapping from learning algorithms to their inductive bias is highly non-trivial. Numerous circuit features (learning rules, architecture, non-linearities, etc.) influence generalisation. For example, training two simple ReLU networks of different depth to classify three data points leads to different generalisations for non-obvious reasons, Fig. 1C. In a few cases analytic bridges have been constructed that map learning algorithms to their inductive bias. In particular, the study of kernel regression, an algorithm that maps data points to a feature space in which linear regression to labels is performed (Bordelon, Canatar, and Pehlevan, 2020; Simon, Dickens, and DeWeese, 2021; Sollich, 1998), has been influential: all the cited examples of understanding in neuroscience via inductive bias have used this bridge. However, it severely limits the approach: most biological circuits cannot be well approximated as performing a fixed feature map then linearly regressing to labels!

Here, we develop a flexible approach that is able to learn the inductive bias of essentially any supervised learning algorithm. It follows a meta-learning framework: an outer neural network (the meta-learner) assigns labels to a dataset, this labelled dataset is then used in the inner optimisation to train the inner neural network (the learner). The meta-learner is then trained on a meta-loss which measures the generalisation error of the learner to unseen data. Through gradient descent on the meta-loss, the meta-learner meta-learns to label data in a way that the learner finds easy to generalise. These easy-to-generalise functions form a description of the inductive bias, since easy-to-generalise functions are those that learners generalise appropriately from few training points. They are therefore functions the network would regularly use to explain finite datasets. In sum, networks are inductively biased towards easy-to-generalise functions.

To our knowledge, the most related work is (M. Y. Li, E. Grant, and Griffiths, 2021). Li et al. view sets of neural networks, trained or untrained, as a distribution over the mapping from input to labels. They fit this distribution by meta-learning the parameters of a gaussian process which assigns a label distribution to each input. This provides an interpretable summary of fixed sets of network. In our work we do something very different: rather than focusing on a fixed, static set of networks, we find the inductive biases of learning algorithms via meta-learning easily learnt functions. We recommend potential users to consider both approaches.

In the following sections we describe our scheme, and validate it by comparing to the known inductive biases of linear and kernel regression. We then extend it in several ways. First, networks are inductively biased towards areas of function space, not single functions. Therefore we learn a set of orthogonal functions that a learner finds easy to generalise, providing a richer characterisation of the inductive bias. Second, we introduce a framework that asks how a given design choice (architecture, learning rule, non-linearity) effects the inductive bias. To do that, we assemble two networks that differ only by the design choice in question, then we meta-learn a function that one network finds much easier to generalise than the other. This can be used to explain why a particular circuit feature is present. We again validate both schemes against linear and kernel regression. Finally we show our tool's flexibility in a series of more adventurous examples: we validate it on a challenging differentiable learner (a spiking neural network); we show it works in high-dimensions by meta-learning MNIST labels; we highlight its explanatory power for neuroscience by using it to normatively explain patterns in recent connectomic data; and we extract inductive biases using gradient-free techniques, demonstrating a method that could be used to extract animals' inductive biases.

11.2 Meta-Learning the Inductive Biases

Our main contribution is a meta-learning framework for extracting the inductive bias of differentiable learning algorithms, Fig. 2A, that we describe in this section. In the outer-loop a neural network, the meta-learner, assigns labels to inputs sampled from some distribution, hence creating the real-world function that our circuit of interest will try to learn. The inner-loop learning algorithm, the learner, is the circuit whose inductive bias we want to extract; for example, a biological sensory processing circuit that assigns valences to inputs. When provided with a training dataset of inputs and labels the learner adjusts its parameters according to its internal

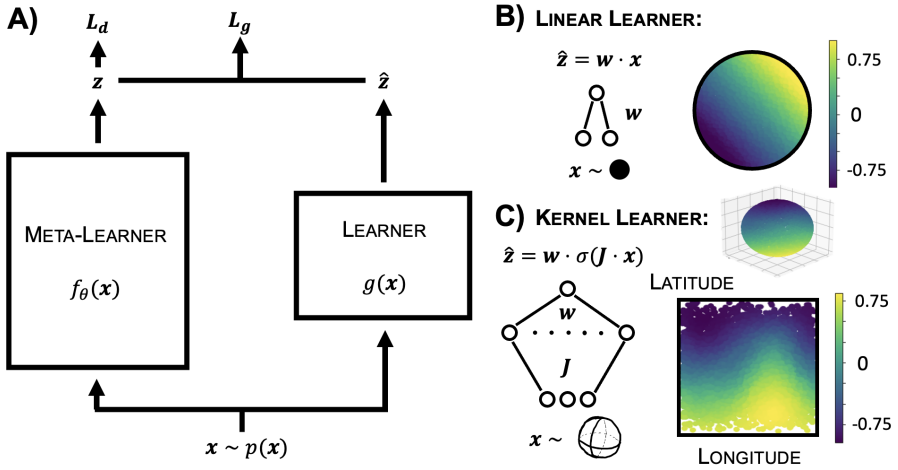


Figure 11.2: **Meta-Learning the Inductive Bias. A:** The meta-learner labels a dataset, part of which is used to train the learner. Gradient descent is performed on a loss made of the learner's generalisation error on unseen data ($\mathcal{L}_g$), and the Sinkhorn divergence between the meta-learner's label distribution and a target distribution ($\mathcal{L}_d$), here choosen to be uniform from -1 to 1. **B:** The meta-learner learns a linearly separable labelling of data sampled from a circle for a ridge regression learner. The plot (and similar future plots) shows the datapoints - the circle - coloured by the assigned meta-learner label. **C:** For a kernel regression learner and data sampled from the surface of a sphere, the meta-learner's labelling is very close to the predicted spherical harmonic (99% of norm within first order harmonics).

learning rules. Then the generalisation error of the trained learner is measured on a held-out test set, and this is used as the meta-loss to train the meta-learner. This process repeats, reinitialising then retraining the learner at every iteration, iteratively developing the meta-learner's weights, until the meta-learner is labelling the data in a way that the learner finds easy to generalise after training on a few datapoints (we used $\sim$30). Thus, the meta-learner has extracted a function towards which the learner is inductively biased.

As just outlined, the meta-learner will find the easiest-to-generalise function, usually the one that assigns all inputs the same label. To avoid this trivial function, we introduce a term in the meta-loss that forces the distribution of labels to take a particular (non-constant) form. Specifically, it measures the Sinkhorn divergence between the meta-learner's label distribution and a uniform distribution from -1 to 1 (other divergences also work, Appendix B, or other methods like forcing the label distribution's variance to be 1). The full pseudocode is in Algorithm 1.

Our meta-learner must learn a function that the learner can generalise. To enable the meta-learner to learn all functions the learner might plausibly generalise well, its function class could usefully be a superset of the learner's. Therefore, we choose the meta-learner's architecture to be a slightly larger version of the learner's (though, beyond this, our findings appear robust, Appendix D).

Just like any other deep learning approach, the meta-learner's hyperparameters (learning rate etc.) must be chosen so that it optimises well. On the other hand, the learner's hyperparameters are chosen as part of specifying the learner - changing the learning rate or initialisation correspond to changing its inductive bias, so the method does and should extract different functions as these parameters vary.

Algorithm 1 Meta-Learning the Learner's Inductive Bias

Require: Initialise meta-learner: $f_\theta(\boldsymbol{x})$
while Step count $<$ Total **do**
  Generate dataset from input distribution: $\boldsymbol{x} \sim p(\boldsymbol{x})$
  Label using metalearner: $z = f_\theta(\boldsymbol{x})$
  Split inputs and labels into test & train sets: $\mathcal{D}_{Tr}$ & $\mathcal{D}_{Te}$
  Train leaner using $\mathcal{D}_{Tr}$ giving trained learner: $g(\boldsymbol{x})$
  Label $\mathcal{D}_{Te}$ using trained learner: $\hat{z} = g(\boldsymbol{x})$
  Compute the learner's generalisation error: $\mathcal{L}_g = \sum_i (z_i - \hat{z}_i)^2$
  Compute the Sinkhorn divergence of metalearner's labels from uniform $[-1, 1]$: $\mathcal{L}_d$
  Take θ gradient step on meta-loss: $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_d$
end while

We validate our scheme by meta-learning sensible functions for linear and kernel learners, whose inductive biases are known. First, for ridge regression on data sampled from a 2D circle the meta-learner assigns linearly

separable labels, Fig. 2B; exactly the labels linear circuits easily generalise.

Next, we meta-learn kernel ridge regression's inductive bias. Kernel regression involves projecting the input data through a fixed mapping to a feature space (e.g. the last hidden layer of a fixed neural network) and performing linear regression from feature space to labels. (Bordelon, Canatar, and Pehlevan, 2020) show that the inductive bias of kernel regression can be understood through the kernel eigenfunctions ($\{v_i(\boldsymbol{x})\}$ with eigenvalue $\{\lambda_i\}$). These are defined on input distribution $p(\boldsymbol{x})$ via a kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ that measures the similarity of two inputs in feature space:

$$\int k(\boldsymbol{x}, \boldsymbol{x}') v_i(\boldsymbol{x}') dp(\boldsymbol{x}') = \lambda_i v_i(\boldsymbol{x}). \tag{11.1}$$

The algorithm is inductively biased towards higher eigenvalue eigenfunctions; i.e., fewer training points are needed to reach a given generalisation error when fitting high vs. low eigenvalue eigenfunctions. General functions can be understood by projecting onto the eigenbasis. Hence our meta-learner, in searching for kernel regression's easiest-to-generalise non-constant function, should choose the highest eigenvalue eigenfunction.

To test this, we meta-learn the inductive bias of a two-layer neural network with fixed first layer weights. We sample data uniformly from the sphere and randomly connect a large hidden layer of ReLU neurons to the three input neurons. The elements of this random weight matrix are drawn *iid* from a standard normal, and the learning algorithm performs ridge regression on the hidden layer activities. Previous work has analytically derived the kernel for this network, and computed its eigenfunctions (Cho and Saul, 2009; Mairal and Vert, 2018), which are spherical harmonics[1]. The higher the frequency of the spherical harmonic the lower its eigenvalue. Matching this, our network meta-learns one of the set of lowest frequency spherical harmonics, Fig. 2C.

11.3 Meta-Learning Areas of Function Space

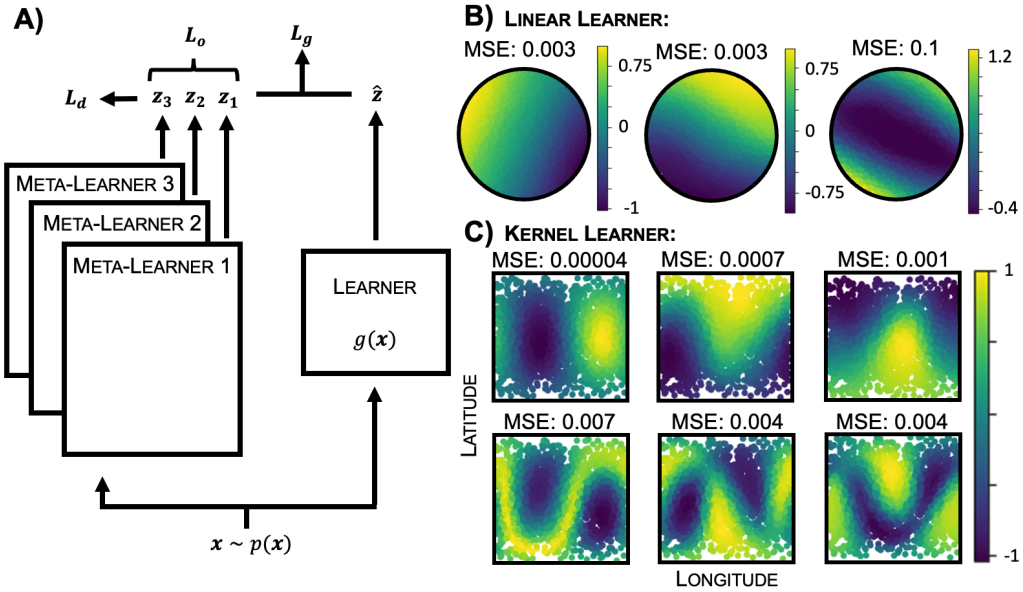


Figure 11.3: **Meta-Learning Many Functions. A:** We learn many meta-learners, each of which has to label orthogonally to all previous meta-learners. **B:** For a linear learner the meta-learners learn two orthogonal linear functions and an orthogonal but hard to learn third function (MSE: Mean Squared Error). **C:** For a kernel learner we learn 6 meta-learners, the first 3 approximate well first order spherical harmonics (96% norm overlap), and the next 3 second order spherical harmonics (91% norm overlap), as predicted.

Having validated our tool on some simple test cases, we now extend it to find a richer characterisation of the inductive bias. A given learning algorithm is inductively biased towards areas of function space, not just one particular function. To gain access to this larger space, we learn a series of meta-learners. The first of these is exactly as described above, then we iteratively introduce additional meta-learners. To ensure each meta-learner learns a new aspect of the inductive bias we add a term to the meta-loss that penalises the square of the dot product between the current meta-learner's labelling and that of all the previously trained meta-learners, Fig. 3A.

[1]Spherical harmonics are an orthonormal basis of functions of increasing frequency defined on the sphere, analgously to sines and cosines on the circle.

On a dataset $\{\boldsymbol{x}_n\}$:

$$\mathcal{L}_{\text{Orthog}} = \sum_i \left(\sum_n f_{\theta_i}(\boldsymbol{x}_n) f_{\theta'}(\boldsymbol{x}_n) \right)^2 \tag{11.2}$$

for each previous meta-learners $f_{\theta_i}(\boldsymbol{x})$ and the current meta-learner $f_{\theta'}(\boldsymbol{x})$. From the learner's perspective nothing has changed, at each meta-step it simply learns to fit the meta-learner that is currently being trained. But each additional meta-learner must discover an easy-to-generalise function that is orthogonal to all previous meta-learners.

We again validate this scheme on linear and kernel regression. For linear regression of 2D data the meta-learners learn two orthogonal linear labellings, then a third orthogonal function that the learner struggles to generalise, as expected, Fig. 3B. For the kernel regression network we described previously theory predicts that the meta-learners should learn the eigenfunctions in decreasing order of their eigenvalue. We find this to a good approximation, Fig. 3C, learning approximations of, first, the three first order spherical harmonics, and second, three linear combinations of the second order harmonics.

For linear classifiers (e.g. linear and kernel regression) the full set of orthogonal functions explains the entire inductive bias, since the average generalisation error on any target function is the sum of the errors on each kernel eigenfunctions weighted by projection of the target function onto that eigenfunction. This will not in general be true, since for a non-linear classifier the generalisation error on a function $f(\boldsymbol{x}) = f_a(\boldsymbol{x}) + f_b(\boldsymbol{x})$ does not equal the generalisation error on $f_a(\boldsymbol{x})$ plus that on $f_b(\boldsymbol{x})$. Nonetheless, we expect the set of orthogonal functions will be a helpful guide to a network's inductive bias, even for non-linear classifiers.

11.4 Isolating a Design Choice's Impact

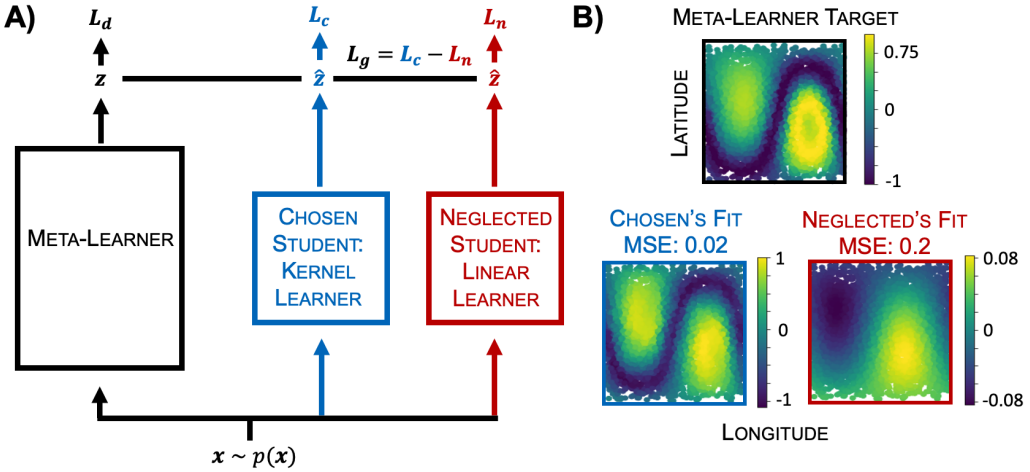


Figure 11.4: **Meta-learning design choice impact. A:** Labellings are learnt so a chosen student generalises but a neglected one doesn't. **B:** The meta-learner finds a non-linear labelling on which kernel regression generalises an order of magnitude better than linear regression.

Our work is motivated by the desire to understand how design choices in learning algorithms - such as architecture or non-linearities - lead to downstream generalisation effects, particularly in biological networks. One additional setting which we have found useful is to compare two networks with some difference between them, and learn functions that one of the networks finds much easier to generalise than the other. In this way, we can build intuition for the impact of design features on the inductive bias. To illustrate this we again create a meta-learner that labels data, but this time the labels are used to train two learners. We then train the meta-learner so that one learner (the chosen student) is much better at generalising than the other (neglected student). This is done by minimising the generalisation errors of the chosen student minus the neglected student, Fig. 4A. Validating this approach on well understood algorithms, we show that it can find functions that a kernel regression algorithm is able to learn better than linear regression, Fig. 4B, i.e. a non-linearly separable function.

This illustrates some of the games that can be played in this setting. For example, you could play a co-operative game, in which you meta-learn a function that a set of learners all find easy to generalise, and each learner could have different connectivity matrices to match the distribution in real animals, ensuring the tool does not over-fit to some specific details. However as the losses become more complex training becomes harder, for example this adversarial setting between chosen and neglected student is hard to make robust if the two learners are relatively similar.

11.5 Applying our Framework

So far we have developed and tested a suite of tools for extracting the inductive bias of learning algorithms. We now apply them to networks whose inductive bias cannot be understood analytically. Specifically: we show our method works on a challenging differentiable learner, a spiking neural network; we validate our method on a high-dimensional MNIST example; we illustrate how our tool can give normative explanations for biological circuit features, by meta-learning the impact of connectivity structures on the generalisation of a model of the fly mushroom body; and we demonstrate a method to extract animals' inductive biases. Our tool is flexible: by taking gradients through the training procedure we can meta-learn inductive biases for networks trained using PyTorch, for example. We share all our code[2] which should be easily adapted to networks of interest.

11.5.1 Spiking Neural Network

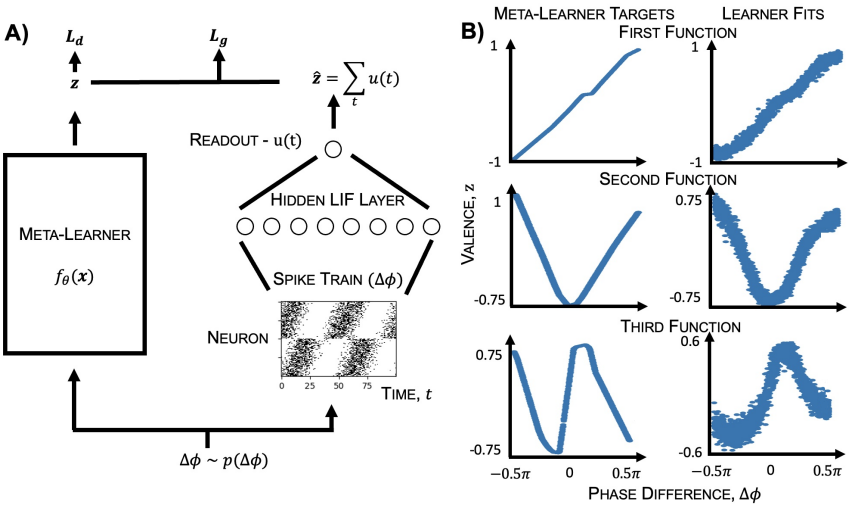


Figure 11.5: **Meta-learning through a Spiking Network. A:** Labellings are learnt that the spiking network, with weights trained via surrogate gradient descent, finds easy to generalise. Phase differences, $\Delta\phi$, are sampled uniformly and used to generate spike trains by sampling from a poisson process with the following rates: for half the neurons $r_n = \frac{r_{\max}}{2}(1 + \sin(t + \theta_n))^2$, where n is a neuron index and θ_n are uniformly sampled offsets; for the other half we add a phase shift: $r_n = \frac{r_{\max}}{2}(1 + \sin(t + \theta_n + \Delta\phi))^2$. These populations represent sensory neurons in the two ears, and $\Delta\phi$ is the interaural phase difference. This activity feeds into a population of linear-integrate-and-fire neurons, then to a readout linear-integrate neuron. The valence assigned is the sum of the readout's activity over time. **B:** We learn three orthogonal meta-learners (as in section 11.3). The spiking network finds it easiest to learn low frequency functions. Left: the meta-learner's target function. Right: the spiking network's labelling. The spiking network captures the main behaviour, but increasingly poorly.

The brain, unlike artificial neural networks, computes using spikes. 'How?' is an open question. A recent exciting advance in this area is the surrogate gradient method, which permits gradient based training of spiking neural networks by smoothing the discontinuous gradient (Neftci, Mostafa, and Friedemann Zenke, 2019; Friedemann Zenke and Vogels, 2021). We make use of this development to meta-learn the inductive bias of a spiking network, providing a challenging test case for our method.

We study a modification of a model developed for a recent tutorial (Goodman et al., 2022; Freidemann Zenke, 2019), which is trained to assign a label to an incoming spike train. The network is a model of an interaural phase difference detection circuit. The input spike train is parameterised by a phase difference, $\Delta\phi$, that generates two sets of spike trains, one in each ear, Fig. 11.5A. These spikes are processed through a hidden layer of linear-integrate-and-fire neurons (LIF), before reaching a classification layer. A real-valued valence is assigned by summing the output neuron's activity over the trial. The meta-learning framework is as before: the meta-learner assigns valences to input phase differences, these labels are used to train the spiking network by surrogate gradient descent, then the meta-learner is trained to minimise the learner's generalisation error and a distribution loss. Our method works well, finding a simple smoothness prior, Fig. 11.5B.

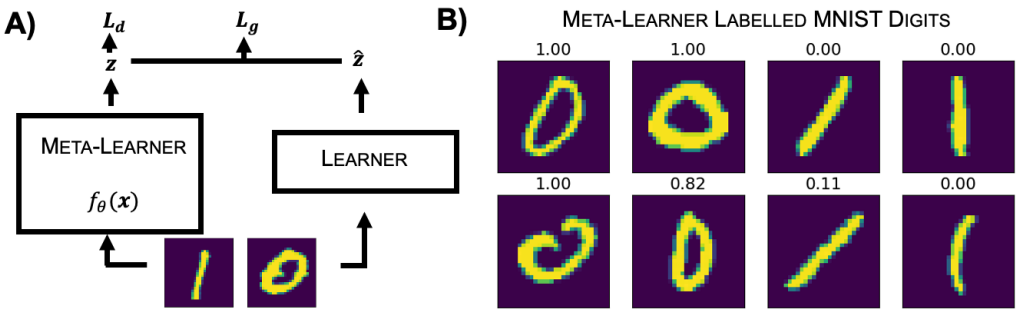


Figure 11.6: **Meta-Learning on MNIST A:** A meta-learner receives MNIST 0s and 1s, and assigns labels, bounded between 0 and 1, that have high variance and can be easily generalised by the learner. **B:** 99% of digits are assigned a label, shown in title, consistent with MNIST class.

11.5.2 A High-Dimensional MNIST Example

Next, we test out method on a high-dimensional input dataset. Thus far, to visualise our results, we have only considered low dimensional input data. We demonstrate that our method continues to work in high-dimensions by applying it to a dataset made of the 0 and 1 MNIST digits (LeCun, 1998). We meta-learn a labelling of this dataset that a simple convolutional neural network finds easy to generalise. Our meta-learner's architecture is also a convolutional neural network whose outputs are bounded between 0 and 1, and the meta-learner must learn an easy-to-generalise labelling with high variance. We find that the meta-learner consistently rediscovers the MNIST digits within the dataset, separating each digit into its own class, figure 11.6.

While successful, this example highlights the major challenge of applying our method in high-dimensions. To properly probe the inductive bias of a CNN would require finding an easily-generalised set of functions from images to labels. How would you intepret such functions? We return to this concern in the discussion; fortunately, we show in the next section that our method can be practically useful when combined with meaningful projections of the data.

11.5.3 Assigning Normative Roles to Connectomic Data

A large maturing source of neuroscience data is connectomics (a list of which neurons connect to one another). However, there is currently a dearth of methods for interpreting this data (Litwin-Kumar and Turaga, 2019). In this section, we show our tool can be used to give normative roles to connectomic patterns through their induced inductive bias. We study a model of the fly mushroom body, a beautiful circuit that fruit flies use to assign valence to odours (Aso et al., 2014; Hige, 2018), for which connectomic data has recently become available (Zheng, Lauritzen, et al., 2018; Zheng, F. Li, et al., 2022).

Odorants trigger a subset of the fly's olfactory receptors. These activations are represented in a small glomerular population (input neurons), projected to a large layer of Kenyon cells (hidden neurons), then onwards to output neurons that signal various dimensions of the odour's valence, Fig. 11.7A. An error signal is provided if the fly misclassifies a good odour as bad, or vice versa, allowing the fly to update its weights and learn appropriate responses. Classically, the input-to-hidden connectivity was assumed random; i.e., each hidden neuron connects to a few randomly selected inputs. However, connectomic data has shown two patterns; first, hidden neurons preferentially connect to some inputs, and second, there are input groupings - if a hidden neuron connects to one member of a group it likely connects to many, Fig 11.7D (Zheng, Lauritzen, et al., 2018; Zheng, F. Li, et al., 2022). (Zavitz et al., 2021) tested networks with this connectivity on a battery of tasks and found that, compared to random, (1) they were better at identifying odours that activated over-connected inputs, and (2) they generalised assigned valence across a group (i.e. if you assign high valence to the activation of one neuron, you do the same for other neurons in the same group).

We used our tool to verify and develop these findings by examining the effect of different connectivity patterns on the inductive bias of a sparsely-connected model of the mushroom body, Fig. 11.7A, Appendix E. As a baseline, fully connected networks are biased towards smooth functions, appendix 11.A.1, the simplest being those that assign valence based on one direction in the input space: high at one end, low at the other, like in Fig. 11.2B - C. However, which direction is unimportant; they're all equally easy to learn. Sparsity breaks this degeneracy, aligning the easiest to learn functions with the input neuron basis, figure 11.7B. As such, sparse connectivity, which is ubiquitous in neuroscience, ensures the fly is best at assigning labels based on the activity of small collections of neurons. Next, we introduced the observed connectomic structure. Biasing the connectivity, so some inputs have more connections than others, broke the degeneracy amongst neuron axes. The networks were, as expected, best at generalising functions that depended on the activity of overconnected inputs, figure 11.7C,

[2]Code at `https://github.com/WilburDoz/Meta_Learning_Inductive_Bias`

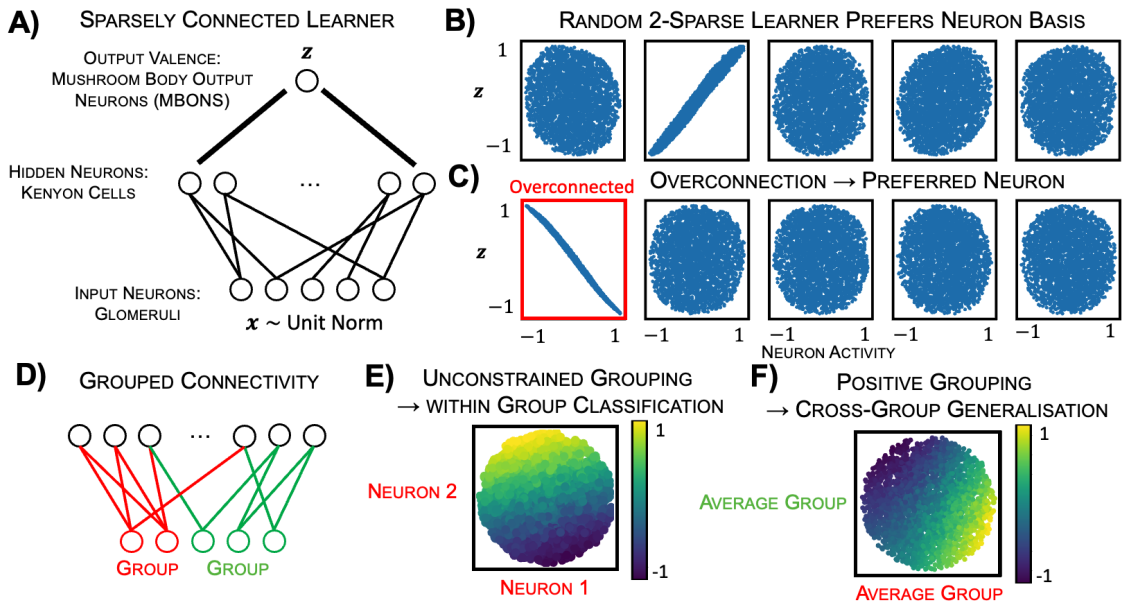


Figure 11.7: **Understanding Connectivity via Inductive Bias. A:** We model the fly mushroom body as a ReLU network with five inputs and one large hidden layer. Each hidden neuron is connected to two of the five input neurons, and we study three networks in which these two connections are chosen differently, either randomly (**B**), biased (**C**) or grouped (**D** - **F**). **B:** Shows the easiest-to-generalise function for a network in which each hidden neuron is connected to two random inputs. Each of the five plots show this function projected against one input neuron's activity. As can be seen, the labelling depends on only one neuron's activity, second from left. **C:** In the overconnected setting each hidden neuron still connects to two inputs, but there is a strong bias towards connecting to the first, highlighted neuron. As a result, the meta-learner settles on a labelling that depends only on this neuron's activity. **D:** We explore the impacts of group connectivity, in which the input neurons are divided into two groups, and hidden neurons tend to be connected to two neurons from the same group. **E:** We train the meta-learner, and find that it's labelling depends only on neurons within the same group. The plot shows the projection of the datapoints into a subspace defined by the two input neurons in the red group. The labelling depends linearly on position within this subspace. **F:** However, if the input-hidden connections are constrained to be positive, the meta-learner's labelling depends only on the average activity within each group, i.e. if one member of a group increases the output, so do all members; hence, the function generalises across group members. The plot shows the two dimensional subspace defined by the mean activity of each of the groups of inputs.

matching (Zavitz et al., 2021). Finally we introduce connectivity groups, figure 11.7D. Without additional changes this does little, the neuron basis is still preferred and, unlike (Zavitz et al., 2021), generalisation across inputs is not observed, figure 11.7E. Only when we additionally constrain the input-to-hidden connections to be excitatory (i.e. positive) do we see that the circuit becomes inductively biased towards functions that generalise across groups of inputs, figure 11.7F. In retrospect this can be understood intuitively: positive weights and grouped connectivity ensure that a hidden neuron that is activated by one input will also be activated by other group members, encouraging generalisation. This effect is removed by permitting negative weights, which let members of the same group excite or inhibit the same hidden neuron.

Thus, we verify and extend the findings of (Zavitz et al., 2021). We, first, show the results hold without needing to hand-design a battery of tasks. This avoids a potential flaw in the approach of (Zavitz et al., 2021): you may reach an incorrect conclusion simply because your battery is not comprehensive enough! (though in this case we agree wholeheartedly with the conclusions of (Zavitz et al., 2021)) Our method avoids this problem by meta-learning the appropriate tasks. Second, we study networks in which both sets of weights are trained, rather than just readout, and, third, we emphasise the role of sparsity in aligning easy-to-learn functions with the neuron basis, something not studied in Zavitz et al. In sum, this example illustrates how our tool can be used to normatively understand elements of circuit design.

11.5.4 Animals' Biases via Black-box Optimisation

In our final, slightly kooky, experiment we modify and test an approach to directly interface with an animal. After all, we are interested in animals' inductive biases and how these arise from neural circuitry; in many settings

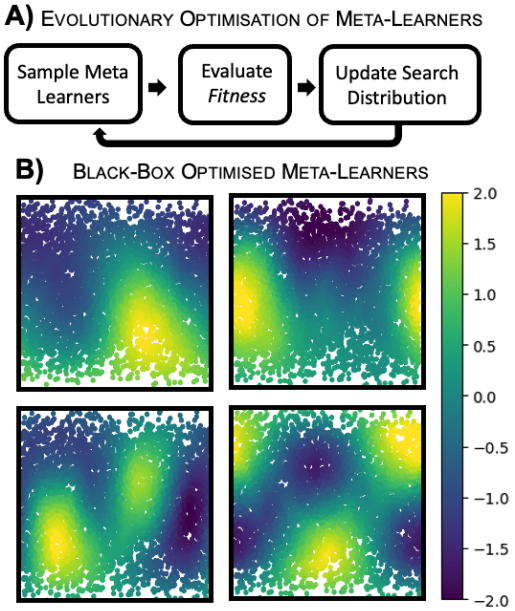


Figure 11.8: **Black-Box Optimisation of Meta-Learners. A:** Evolutionary optimisation of meta-learners involves sampling a set of meta-learner parameters, evaluating their fitness, and using these evaluations to update the search distribution (using a scheme described in (Lange et al., 2022)). The fitness is the full loss, including the learner's generalisation error, the orthogonality loss (eqn. 2), and a label distribution loss. **B:** Trained on the kernel regression problem from section 2 it recovers functions of increasing frequency, though misses one of the first order spherical harmonics.

a better way to probe these is to avoid modelling the circuit entirely, and to simply extract easy-to-generalise functions from the animal's behaviour. This can be done by replacing the inner learner with a real animal trained on a labelled dataset from the meta-learner, then tested on new datapoints. This quickly runs into a problem: we cannot take gradients through the computations of a living animal. However, we can avoid needing these gradients by using evolutionary optimisation to train the meta-learner, which requires only evaluations of the generalisation error.

In figure 8 we demonstrate the efficacy of this scheme. We use the Discovered Evolution Strategy from (Lange et al., 2022) (implemented in evosax (Lange, 2022)), to learn functions that the simple kernel ridge regression circuit discussed in section 2 generalises well. We sample a distribution of meta-learners, measure the learners generalisation error on these, and update the meta-learner distribution accordingly. We are eager to try this approach experimentally.

11.6 Discussion & Conclusions

We presented a meta-learning approach to extract the inductive bias of differentiable supervised learning algorithms, which we hope will be useful in normatively interpreting the role of features of biological networks. This approach required few assumptions beyond those that make the inductive bias an interesting way to conceptualise a circuit in the first place. We required, first, that the input data distribution was specified, and, second, the circuit must be interpretable as performing supervised learning. We will discuss these requirements and ways they could be relaxed; regardless, it is heartening that any circuit satisfying these will, in principle, suffice. The analytic bridge between kernel regression and its inductive bias (Bordelon, Canatar, and Pehlevan, 2020; Simon, Dickens, and DeWeese, 2021) has already found multiple uses in biology in just a few years (Bordelon and Pehlevan, 2022; K. D. Harris, 2019; Pandey et al., 2021; M. Xie et al., 2022), despite its stringent assumptions. We hope that relaxing those assumptions will offer a route to allow these ideas to be applied more broadly.

The first requirement is access to an input distribution, which is often lacking. This can be avoided by using real neural data as the input. Or, if neural data is limited, generative modelling could be used to fit the neural data distribution and new samples drawn from that distribution. Finally, one could imagine a single meta-learner that creates not only the label, but also the data. That is, the meta-learner could generate the entire dataset by transforming a noise sample into an input-output pair. This would have to be carefully regularised to avoid trivial input distributions, but could in principle learn the input statistics that particular networks are tuned to process.

Next, we could relax our second assumption, that the learner performs supervised learning. This is a standard assumption in theoretical neuroscience (Hiratani and P. E. Latham, 2022; Schaffer et al., 2018; Sorscher, Ganguli, and Sompolinsky, 2022), and is often reasonable. Some circuits contain explicit supervision or error signals, like the fly mushroom body or the cerebellum (Shadmehr, 2020), and generally brain areas that make predictions (i.e., all internal models), can use their prediction errors as a learning signal. Alternatively, some circuits are well modelled as one area providing a supervisory signal for another, as in classic systems consolidation (McClelland, McNaughton, and O'Reilly, 1995), or receiving supervision from a past version of themselves through replay (Ven, Siegelmann, and Tolias, 2020). Nevertheless, much biological learning seems to be unsupervised. Our framework could be extended to these settings by assuming an unsupervised objective and meta-learning the dataset on which an observed circuit performs well. For example, the unsupervised objective might be to produce a lower-dimensional representation of the data with the same dot-product similarity structure as the inputs. If the learner was doing PCA, then the meta-learner would learn data that lay in a linear subspace. It would be interesting to try this for different dimensionality reduction algorithms, or to see which bits of input structure biological unsupervised networks are tuned to process.

Despite our optimism for this approach, there remain challenges. Most fundamentally, arbitrary functions are still hard to interpret. Our tool turns one problem - what is the inductive bias of this learner - into another - can I intepret this function the learner finds easy-to-generalise? The second problem is hard but tractable, and is an active area of research (Montavon, Samek, and K.-R. Müller, 2018); and we hope progress in this area will help our tool. In the meantime, we have shown how our tool provides insight in a variety of settings: for low-dimensional inputs (Fig. 2 - 5), by comparing to ground truth labels (Fig. 6), or by projecting the learnt functions onto an appropriate basis (Fig. 7).

To conclude, the inductive bias is a promising angle from which to understand learning algorithms. Analytic bridges between circuit design and inductive bias have already 'explained' the presence of aspects of the circuit through their effect on the network's generalisation properties in both artificial (Bahri et al., 2021; Canatar, Bordelon, and Pehlevan, 2021) and biological (Bordelon and Pehlevan, 2022; K. D. Harris, 2019; Pandey et al., 2021; M. Xie et al., 2022) networks. However, these techniques require very constraining assumptions. We have dramatically loosened these assumptions and shown our tools utility in, among other things, interpreting connectomic data. We believe it will prove useful on other datasets and problems.

11.A Technical Appendices: Meta-learning Inductive Biases

11.A.1 Meta-Learning a Simple Backprop Trained ReLU Network

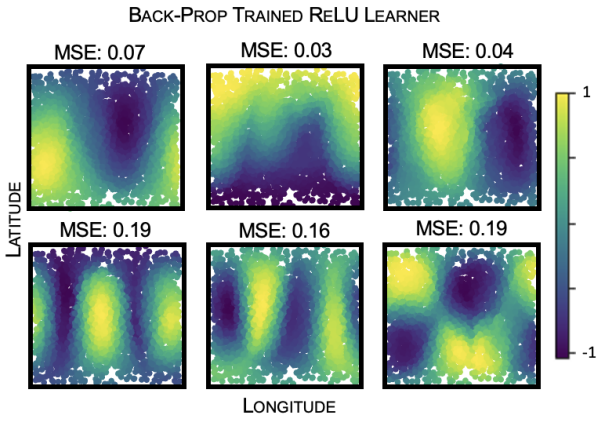


Figure 11.9: **Meta-learning a Backprop Trained Network** We meta-learn the functions a 2-layer ReLU network trained using backprop finds easy to generalise, as can be seen, this shows a low-frequency bias.

A simple test of our framework is a feedfoward ReLU network with 2 hidden layers, learnt using gradient descent. While the functions this network finds easy to generalise cannot be extracted analytically, our tool finds that, unsurprisingly, these networks are biased towards smooth explanations of the data, learning six smooth orthogonal classifications that increase in frequency and mean squared error Fig. 11.9B. We include this example as the simplest PyTorch implementation of learning the inductive bias of a network trained by gradient descent, in the hope that the code can be easily adapted for future use.

11.A.2 Using Different Divergence Measures

To persuade the meta-learner to find non-trivial functions, we include a divergence loss that forces the meta-learner's label distribution to take a particular form: uniform between -1 and 1. In this section we show that the particular divergence that we use has little impact on the solutions we find for learning the easiest-to-generalise function of the kernel learner, figure 11.2C. Figure 11.10 shows that a variety of divergence metrics can be used, sinkhorn, an energy statistic from (Székely and Rizzo, 2013), and the maximum mean discrepancy from (Gretton et al., 2012) (implementations from (Djolonga, 2020)). In each case the learnt function has over 99% norm in the space of first order spherical harmonics, demonstrating that the meta-learner has learnt appropriately.

Other techniques also work well, including adding terms to the loss that force the variance of the meta-learner's label distribution to be 1, which is what we used in figure 8.

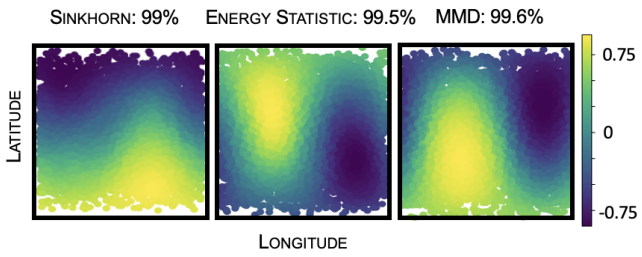


Figure 11.10: We use three different divergence metrics to penalise the meta-learner's label distribution for deviating from uniform between -1 and 1. We add these to the meta-loss, along with the generalisation error of the simple kernel learner introduced in section 2. For all three divergence metrics the meta-learner learns a very close approximation to a first order spherical harmonic, as predicted.

11.A.3 Random Re-Seeding cannot Replace Orthogonalisation

We introduced the orthogonalisation procedure in section 11.3 so that successive meta-learners have to explore larger areas of function space. It is legitimate to wonder whether simply re-running the optimisation would have had the same effect. Here we show it does not, and orthogonalisation is needed to explore the learner's inductive bias fully.

We re-run the meta-learner optimisation without any orthogonality terms for the kernel learner described in section 2. In figure 11.11 we find that the meta-learners find different functions, but only approximations to the first order spherical harmonics. This makes sense, the meta-learner is tasked with finding the easiest-to-generalise non-constant function. For this particular learner there is a degenerate space of such functions and so re-running the meta-learner simply draws another sample from this space of functions. However, to access the second order spherical harmonics that this learner still learns, just worse, we need something like the orthogonality constraint.

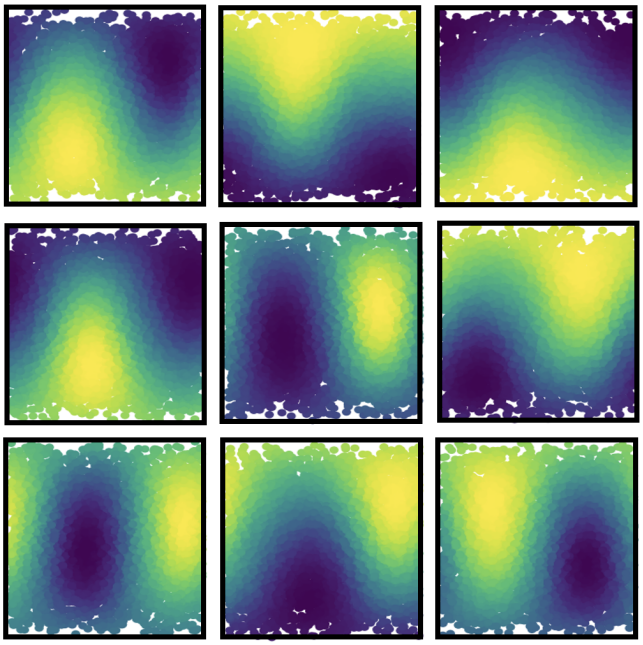


Figure 11.11: Re-running the meta-learner 9 times on the kernel regression algorithm from section 2 produces approximations to the first order spherical harmonics, but no second order functions.

11.A.4 Impact of Meta-Learner Architecture on Extracted Functions

We chose the meta-learner's architecture to be a slightly larger version of the learner's. Our motivation for this is that we want the function class of the meta-learner to be a super-set of the learner's, so that it can learn all the functions the learner could plausibly generalise well.

We tested how robust our results were to architectural changes in the meta-learner. We used the simple feedforward 2-hidden layer ReLU network as in Appendix A, trained by backpropagation, and learnt 6 orthogonal functions that the learner finds easy to generalise. We took our meta-learner to be a similar feedforward network of different depth from 4 to 0 hidden layers. Figure 11.12 shows that the specific choice of meta-learner didn't matter for meta-learners with between 2 to 4 layers. Each learnt 3 approximations to first order spherical harmonics, and 3 to second order harmonics. But a linear learner can't learn more than three orthogonal functions, so failed to find more than three orthogonal generalisable functions.

11.A.5 Connectomic Details

The networks studied in section 5.3 all have 5 input neurons, a large hidden layer of 150 neurons, and a readout neuron. We train the networks end-to-end to minimise the mean square error with one key architectural difference. Each network is constrained to have a sparse input-to-hidden connectivity: each hidden neuron is connected to only two input neurons, these weights are allowed to vary, all other input-to-hidden weights remain fixed at 0. Figure 7 shows results from three networks that all differ only in how each hidden neurons' connections to the input neurons are chosen. In the simplest network (Random 2-Sparse, Figure 7B) these are chosen randomly; each hidden neuron is equally likely to connect to any pair of inputs. For the second model

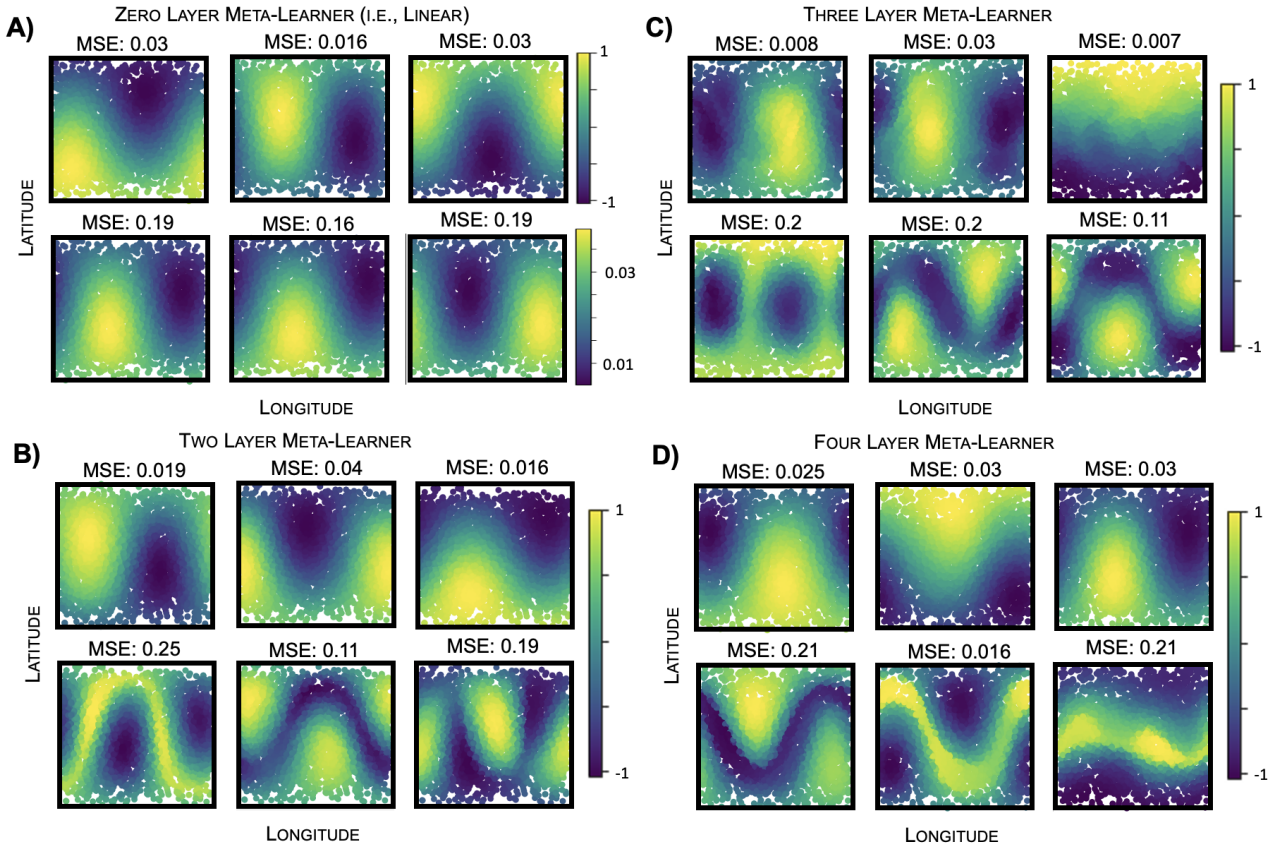


Figure 11.12: We meta-learn 6 easy-to-generalise functions for a simple ReLU network using meta-learners of different depths. This process fails to find more than 3 orthogonal functions when the meta-learner is linear (A.) (note the small label spread, a symptom of a failure to learn), but meta-learners with (B.) 2, (C.) 3, or (D.) all find qualitatively similar results

(Figure 7C) we introduce a bias in the connectivity, one input is much more likely to be chosen than the others, the samples are drawn without replacement from a categorical distrubtion with values 0.6, 0.1, 0.1, 0.1, 0.1. Finally for the last model we introduce grouped connectivity: the inputs come in two groups, if a hidden neuron is connected to one member of a group it is likely to be connected to another. We implement this by dividing the five input neurons into one group of two, and another of three. Each hidden neuron's first connection is drawn randomly from the five. Then the second connection is chosen differently depending on the group of the first: if it connected to a member of the first group with two embers the second connection is chosen with probabilities 0.85, 0.05, 0.05, 0.05; else it is chosen with probabilities 0.1, 0.1, 0.4, 0.4.

Chapter 12

Mapping Structure to Function(s) in Cerebellar-like Networks using Kernel Regression

Abstract

Cerebellar-like networks, in which input activity patterns are separated by projection to a much higher-dimensional space before classification, are a recurring neurobiological motif, present in the cerebellum, dentate gyrus, insect olfactory system, and electrosensory system of the electric fish. Their relatively well-understood design presents a promising test-case for probing principles of biological learning. The circuits' expansive projections have long been modelled as random, enabling effective general purpose pattern separation. However, electron-microscopy studies have discovered interesting hints of structure in both the fly mushroom body and mouse cerebellum. Recent computational work has sought to explain this non-random connectivity through the way it enables the circuit to prioritise learning of some, presumably natural pertinent, tasks over others. Here, we present an alternative route to reach the same conclusions. We build a simplified kernel regression model of the system and use recently derived theoretical results to examine how connectivity impacts the networks ability to learn. We find that the reported structure in the projection weights shapes the network's inductive bias in intuitive ways, making some functions easier to learn and others harder. Our approach is analytically tractable and pleasingly simple. Here we aim to didactically present the technique, in the hope that it may be used to understand the functional implications of other processing motifs in cerebellar-like networks.

12.1 Introduction

The crystalline nature of cerebellar cortex has attracted theoretical work for over 50 years, since it was first posited to perform pattern separation (Albus, 1971; Ito, 1972; Kawato et al., 2021; Marr, 1969). The basic processing involves a feed-forward passage of information from a small number of input mossy fibres, through an expansion to a vastly larger population of granule cells which activate sparsely, and themselves converge onto a small number of output purkinjie cells (Shadmehr, 2020), fig. 12.1B. Versions of this expand-sparsify-contract circuit motif have since been recognised in the dentate gyrus (Borzello et al., 2023), the mushroom body of the fly (Modi, Shuai, and G. C. Turner, 2020), and the electrosensory organ of the electric fish (Bell, 1981). The classic Marr-Albus-Ito model has understood these circuits via pattern separation. Through the expansive and sparse mapping to the granule cell layer, potentially overlapping inputs patterns are separated, allowing supervised learning on the granule-to-purkinjie cell weights to produce the correct output for each input. To first order, this pattern separation machine remains a good model.

However, recent data have driven updates to this theory. For example, it has been found that some representations in both the electrosensory organ and the cerebellar granule cells are not sparse (Badura and De Zeeuw, 2017; Knogler et al., 2017), conflicting with the pattern separation model. Theoretical work has instead argued that variable sparisty levels are a reflection of differing functional roles: very sparse codes are good for categorisation, as in the original Marr-Albus-Ito model, while denser codes are optimal for more continuous prediction tasks, such as forward modelling of body movements (M. Xie et al., 2022), a well-publicised role of cerebellum (Wolpert, R Chris Miall, and Kawato, 1998).

A similar and ongoing update comes from major technical advances in the availability of connectomic data, information about which neurons are connected. Recently connectomic data for both the fly mushroom body

(Zheng, F. Li, et al., 2022) and a section of mouse cerebellum (Nguyen et al., 2023) have become available, and have led to questioning of another aspect of the classic model. Previous work assumed that the connectivity between input and expansion layer is random (Litwin-Kumar, K. D. Harris, et al., 2017; Marr, 1969), and this choice can be justified by the way it produces high-dimensional representations, perfect for pattern separation. The connectomic data, however, contains signs of weak structuring. Rather than connecting randomly to input neurons, expansion layer neurons connect preferentially to some inputs, and, further, the inputs appear to be grouped such that if an expansion neuron is connected to one input it is very likely to be connected to other inputs in the same group, fig. 12.3. In the mushroom body this connectivity structure arises independently of neural activity (Hayashi et al., 2022), suggesting it is genetically hard-wired. We are then prompted to ask, why is the nervous system investing effort to establish this connectivity? What functional role does it play in the pattern separating circuit?

Two recent modelling works have tackled exactly this question. Zavitz et al., 2021 built models of the fly mushroom body with and without the observed structured connectivity motifs. They then trained and tested each network on a battery of tasks, and found that the networks with the observed connectivity performed better at some, presumably naturally pertinent, tasks, and worse at others. In particular, over-connecting to some inputs allowed easy identification of single odours that activated that input, while input groupings made the network better at classifying odours that activated all input neurons within a group, tasks the fly presumably has to perform regularly. William Dorrell, Yuffa, and P. Latham, 2023 instead meta-learned the input-output maps that the circuit found easy to learn with or without the observed connectomic patterns. From this they derived similar conclusions without the need to pre-specify a battery of tasks to test the network on, instead allowing a search from the full space of tasks for those the network performs best at.

In this work we present an alternative route to reach broadly the same conclusions about the functional effect of structured connectivity. Our approach will make use of an established correspondence between cerebellar-like circuits and an algorithm called kernel regression (K. D. Harris, 2019; M. Xie et al., 2022). Recent advances in machine learning theory have characterised the generalisation properties of kernel regression; i.e. how well the algorithm is able to capture input-output mappings from limited training examples (Bordelon, Canatar, and Pehlevan, 2020; Simon, Dickens, and DeWeese, 2021; Sollich, 1998). We will apply these theoretical advances to a cerebellum-like model to understand the effect of different connectivity patterns on the generalisation properties of the network. Using this link we assign a normative role to the structured connectivity: allowing the circuit to learn some, presumably important, classifications from less experience than if the connectivity had been random. Each of the three approaches has their pros and cons. By presenting our approach pedagogically we hope both to broaden the use of inductive bias to understand cerebellar-like circuits, and allow users an informed choice between methods.

We begin in section section 12.2 by describing the structure of cerebellar-like circuits and how they relate to kernel regression. In section section 12.3 we didactically outline theoretical results on kernel regression, and in section 12.4 we use these to analytically understand the impact of connectivity on the network's behaviour. Finally, in section 12.5 we show that these core results generalise to less schematised models.

12.2 Cerebellar-like Networks as Kernel Regression

We begin by outlining our simplified cerebellar-like model and its relationship to kernel regression.

Cerebellar-Like Circuits As the main target for both connectomic work and our investigations, we will describe the fly mushroom body, fig. 12.1A, before later relating it to other cerebellar-like networks. Odours arrive at the periphery of the fly olfactory system and activate a panel of olfactory receptor neurons (ORNs) through interactions with their receptor proteins (Modi, Shuai, and G. C. Turner, 2020). There are many different types of receptor proteins, but each ORN has only one (the different coloured neurons in fig. 12.1A). The axons of ORNs containing each type of receptor converge onto a single glomerulus, of which there are about 50 in total. In the glomerulus, the olfactory receptor neurons synapse with projection neurons (PNs) that carry the information onward towards both the lateral horn, another site of more hard-coded olfactory processing, and the mushroom body. In the mushroom body they pass information onto the 2000 kenyon cells (KCs), which form a large sparse representation of the odours that will serve as the basis for classification. Each KC receives input from roughly 7 projection neurons and a large inhibitory neuron synapses onto all the KCs and ensures that only 5-10% of neurons are active at any one time. Finally, mushroom body output neurons (MBONs) connect to the entire KC population and are thought to represent an odour classification of sorts, signalling various dimensions of the valence of a given odour. Most of the synapses in this circuit are not thought to change on short timescales, barring the important site of learning, the KC-MBON connections. Changes in these synapses are governed by dopaminergic neurons, whose activities are thought to encode errors in recent classifications. Hence, this system forms a neat odour classification device, schematised in fig. 12.1A.

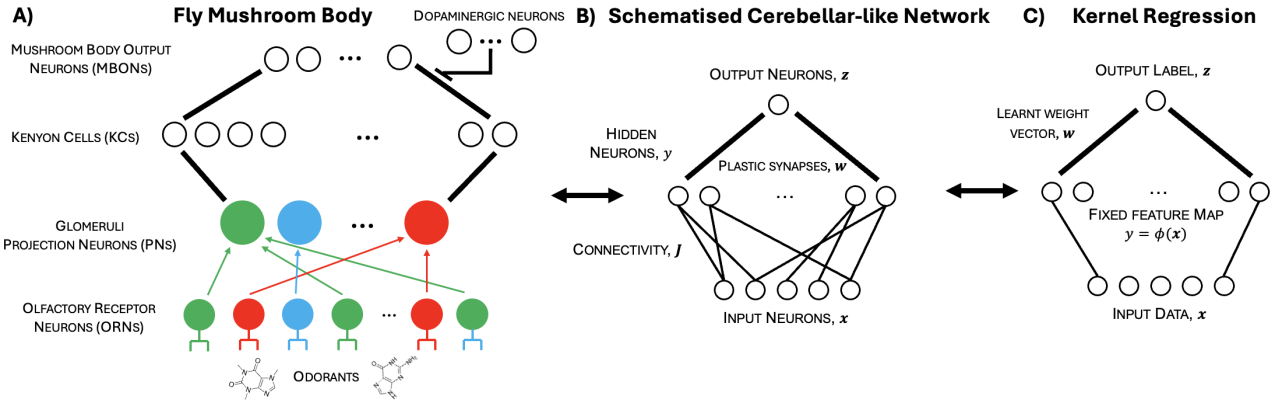


Figure 12.1: **A) - Schematic of fly mushroom body circuit.** Odorants trigger activity in olfactory receptor neurons (ORNs). ORNs contain a unique receptor protein, signalled by their colour, neurons with the same receptor protein send projections to a shared glormulus. There they synapse with projection neurons, which carry the activity to the mushroom body. In the mushroom body the dimensionality of the representation is expanded by a factor of 50 in the kenyon cells (KCs), before being assigned a valence by mushroom body output neurons (MBONs). Dopaminergic neurons encode an error signal that modulates the KC-to-MBON connectivity to ensure correct classification. **B - Simplified Network** We use a model focusing only on the expansive projection and subsequent labelling which, **C**, has uncanny similarities with the kernel regression algorithm.

This design is an example of the broader class of cerebellar-like circuits. The core features that define this circuit motif are the expansive non-linear projection, which serves to separate and pre-process the inputs, and the subsequent downstream classification via plastic synapses modulated by an explicit error signal. Unsurprisingly, this structure is cleanly exhibited by the cerebellum itself, where mossy fibres, granule cells, and purkinjie cells play the roles of input, expansive, and output layers - with an estimated 30 times more granule cells than mossy fibres. Each purkinjie cell has an associated climbing fibre which carries an error signal: if the climbing fibre activates it triggers a complex spike in the purkinjie cell which leads to plasticity at the granule-to-purkinjie synapses, teaching the system to correctly classify inputs. Somewhat similar patterns are seen in the electrosensory lobe of the mormyrid fish (Kennedy et al., 2014), and the dentate gyrus (Borzello et al., 2023), though in all cases there remain significant puzzles overlooked by the simple story presented here (Cayco-Gajic and Silver, 2019).

Simplified Model In this work we are interested in how structure in the expansive projection affects a cerebellar circuit's ability to learn. We therefore use a simplified model that targets this question directly. All assumptions are made for analytic ease, and some of the most egregious will be relaxed in later sections where we'll show our conclusions generalise to a more realistic model.

Our simple model contains only an expansive projection and nonlinearity, followed by a linear classification. We assume the input activity patterns, corresponding to the activity of glomeruli in the mushroom body or mossy fibres in the cerebellum, are sampled uniformly from the surface of a sphere. In this section we'll use only three input dimensions for visualisation, but our analytic results generalise trivially to higher dimensions. This activity is then projected to a much larger population through a connectivity matrix, $\boldsymbol{J}$, each element of which encodes the connection strength between one pair of input and hidden layer neurons. An elementwise non-linearity, ϕ, is applied to produce the final expansive layer activity - $\boldsymbol{y} = \phi(\boldsymbol{J} \cdot \boldsymbol{x})$. We will get some analytic insight using a simple ReLU nonlinearity, but will show the same conclusions hold using a more biologically plausible sparsification so that only $5 - 10\%$ of neurons are active at one time, as observed in the fly mushroom body (Lin et al., 2014). Finally, a readout weight, $\boldsymbol{w}$, assigns a label to each pattern: $\hat{z} = \boldsymbol{w} \cdot \boldsymbol{y}$ and is learnt using a training set of 'true' input-label pairs: $\mathcal{D} = \{\boldsymbol{x}_i, z_i\}$, gathered during the animal's recent experience.

This model matches directly onto a machine learning algorithm called kernel regression, which also comprises a non-linear fixed processing step, followed by a linear classification, fig. 12.1C. This correspondence will let us harness significant theoretical work to understand the impact of connectivity choices on learning.

12.3 Connectivity's impact on Learning - Intuition from Kernels

This work seeks to provide an intuitive yet rigorous exploration of the impact of connectivity structure on learning in cerebellar-like networks. To this end, we will slowly develop some existing ideas, concluding with the result that the generalisation properties of kernel regression are summarised by the kernel eigenfunctions, which

we'll later relate to the choice of connectivity. Readers familiar with kernel regression will likely be bored, but it is hoped that the intuitive description will broaden the applicability of these approaches.

Algorithmic Background The game we're playing is supervised learning: using a set of examples to learn to predict the correct output for any input. Perhaps the simplest widely used supervised learning system is linear regression, where the predicted output, $\hat{z} \in \mathbb{R}$, is a linear function of the input, $\boldsymbol{x} \in \mathbb{R}^N$: $\hat{z} = \boldsymbol{w}^T \boldsymbol{x}$. The weights, $\boldsymbol{w} \in \mathbb{R}^N$, are learnt from training data in a way that hopefully generalises to unseen 'test' data. This is delightfully simple, can be well understood theoretically, and forms the bedrock of all regression analyses; however it suffers from a clear shortcoming: not all outputs are linearly related to their inputs, fig. 12.2A. How can we model biology's ability to learn such non-linear relationships between inputs and outputs? One now-pervasive answer is artificial neural networks, which through iterated layers of non-linear mappings can learn sophisticated mappings. Unfortunately, these networks are not well understood, nor is it clear how learning across such deep architectures could be implemented in the brain (T. P. Lillicrap et al., 2020).

Kernel regression, sitting between these two extremes, suffers from none of the above flaws. As discussed, kernel regression involves first projecting the input data through a fixed non-linear mapping, called a feature map, then linearly classifying the features (rather than the inputs directly). By choosing an appropriate feature map, nonlinear classifications can be performed, fig. 12.2B; indeed, for many choices of feature maps every possible input-output function can be learnt (Micchelli, Yuesheng Xu, and H. Zhang, 2006). Yet kernel regression balances this flexibility with both analytic tractability and biological plausibility - with an uncanny match to cerebellar-like systems. Learning only happens at the output weights and could be driven by a simple error signal - a motif that matches findings across cerebellar-like networks where error signals often drive plasticity at expansion-to-output synapses[1]. Further, its relative simplicity will allow us to precisely understand the impact of connectivity on learning.

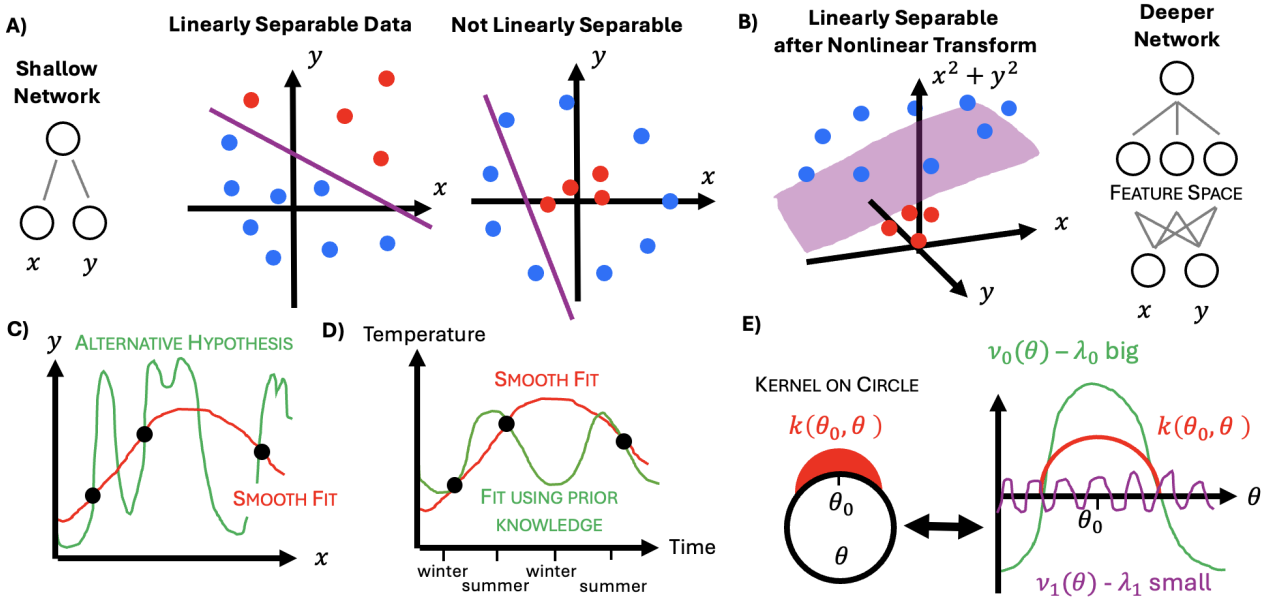


Figure 12.2: **A)** A single linear layer can only classify linearly seperable data. **B)** However adding a fixed first layer of nonlinear processing can permit a linear readout layer to perform nonlinear classifications. **C)** Given any finite dataset there are infinitely many possible generalisations to unseen data. Without prior assumptions there is no reason to choose between them, i.e. from training data alone both curves are equally reasonable. **D)** Generic prior assumptions might prefer a smooth solution, while prior knowledge on how temperature varies throughout the year would select a solution with a period of 12 months. **E)** Left: A kernel is a measure of similarity between points, for example the points could live on a circle and the kernel might be largest for neighbouring points and small at a distance. Right: The weighted sum $\int k(\theta, \theta')v_i(\theta')d\theta'$ is large when the eigenfunction has similar structure to the kernel, green, and small in other cases, purple.

Inductive Bias Having chosen our learning algorithm and established that, given infinite training data, it is capable of learning many non-linear input-output mappings, we are left with a more practical question: in the

[1]In the mushroom body dopamine drives learning at kenyon cell to mushroom body output neurons (Waddell, 2013); in the cerebellum climbing fibres via complex spikes drive learning at granule cell to purkinjie cell synapses (D'Angelo, 2014); and in the electrosensory lobe of the mormyrid fish broad spikes drive plasticity (Bell, Caputi, and K. Grant, 1997; Bell, Caputi, K. Grant, and Serrier, 1993)

realistic setting with relatively small amounts of training data will it ever learn correctly?

Kernel regression, like all learning algorithms, carries with it an inductive bias - a preference for learning certain functions over others. During learning the network adapts to fit a few labelled training examples; however, having fit a finite dataset there are infinitely many ways to generalise to unseen examples, fig. 12.2C, and choosing amongst them without prior assumptions is impossible (D. Hume, 1748). Yet, somehow, the kernel regression algorithm chooses. How it does so is its inductive bias, an embodiement of the algorithm's implicit prior assumptions about the world. A learning algorithm will only be useful if it can learn the functions it is likely to need using a reasonably small amount of data, in other words, if its prior assumptions about the world are reasonable.

A simple illustrative example is a bias towards smoothness, built into almost all learning algorithms. Consider the dataset in fig. 12.2C. In reality it might have come from a high frequency function, or any of the infinitely many other functions that interpolate the training data. But, having observed only a couple of datapoints, it would seem unreasonable to guess a fast-oscillating output. Rather, without any more information, the best choice would seem to be a smooth low-frequency interpolation between the training data. If, however, I told you that the input was time and the output was temperature in an unknown city, your prior assumptions would be shifted, and you might instead predict an output with a period of 12 months, fig. 12.2D.

Kernels The inductive bias therefore becomes a way to understand a learner's behaviour. The link between cerebellar-like networks and kernel regression is appealing because the inductive bias of kernel regression can be precisely understood using the kernel function. The *kernel* in kernel regression is a function that measures the similarity between two inputs. In our context it refers to the similarity between two inputs after being mapped through the non-linear feature map of the cerebellar-like expansive projection:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{J}\boldsymbol{x})^T \phi(\boldsymbol{J}\boldsymbol{x}') \tag{12.1}$$

In more neuroscience language this corresponds to the representational similarity of two inputs in the expansion layer. This is the key quantity both generally for kernel algorithms, and for our current discussion. Just as different cerebellar-networks are specified by their choice of non-linear projection, different kernel regression algorithms are defined by their choice of kernel.

Crucially, the kernel function is the key quantity for determining how easy a function is to learn (i.e. how many training points a network needs to see before it can generalise a function with a given level of accuracy). Kernel regression is inductively biased towards assigning similar outputs to inputs it deems similar. To repeat this vital point, if two inputs, $\boldsymbol{x}$ and $\boldsymbol{x}'$, are represented similarly in the feature map (meaning $k(\boldsymbol{x}, \boldsymbol{x}')$ is large) then the kernel regression algorithm finds it easier to learn a function which assigns them similar labels. We can understand this by imagining a network trained to correctly classify a single training example $\{\boldsymbol{x}_1, z_1\}$. Kernel regression, just like more biologically plausible online regularised delta rule algorithms, learns the min-norm weight vector that correctly classifies this point:

$$\text{Correct Classification:} \quad \boldsymbol{w}^T \phi(\boldsymbol{J}\boldsymbol{x}_1) = \boldsymbol{w}^T \boldsymbol{y}_1 = z_1 \qquad \rightarrow \qquad \text{Min-norm Weight Vector:} \quad \boldsymbol{w} = \frac{z_1 \boldsymbol{y}_1}{||\boldsymbol{y}_1||^2} \propto z_1 \boldsymbol{y}_1 \tag{12.2}$$

The resulting weight vector aligns with the datapoint's expansion-layer representation, $\boldsymbol{y}_1$. This means that when presented with unseen test points the system will generalise what it has learnt in a manner that depends exactly on the similarity between the test and training point. We can see this by calculating the label assigned by such a weight vector to any new input, and seeing that it depends explicitly on the kernel:

$$\hat{z}(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{J}\boldsymbol{x}) \propto z_1 \phi(\boldsymbol{J}\boldsymbol{x}_1)^T \phi(\boldsymbol{J}\boldsymbol{x}) = z_1 k(\boldsymbol{x}_1, \boldsymbol{x}) \tag{12.3}$$

Similar ideas generalise to networks trained on multiple datapoints. In summary, any network must choose how to generalise what is has learnt from training examples. Kernel regression chooses its generalisation according to its inbuilt similarity measure, the kernel function: $k(\boldsymbol{x}_1, \boldsymbol{x})$. If the true input-output function, $z(\boldsymbol{x})$, matches the structure assumed by the kernel then this generalisation will be appropriate, and the network will require few training examples to achieve low generalisation error. If, however, the algorithm's assumptions are wrong, for example perhaps the true function really is periodic with a period of 12 months while the kernel is just says two timepoints are similar if they are nearby, then the network will generalise to unseen regions poorly.

Theoretical Results Pleasingly, theoretical work has made the preceeding logic precise (Bordelon, Canatar, and Pehlevan, 2020; Simon, Dickens, and DeWeese, 2021; Sollich, 1998). These works characterise the generalisation properties of the network in terms of the kernel's eigenfunctions, $v_i(\boldsymbol{x})$, and their eigenvalues, λ_i, defined, given some input distribution $p(\boldsymbol{x})$, analogously to a continuous version of eigenvectors:

$$\int k(\boldsymbol{x}, \boldsymbol{x}') v_i(\boldsymbol{x}') dp(\boldsymbol{x}') = \lambda_i v_i(\boldsymbol{x}) \tag{12.4}$$

We can understand this characterisation using the following fact: if an eigenfunction has a high eigenvalue it assigns similar values to inputs that the kernel deems similar. This can be seen by viewing the eigenfunction, $v(\boldsymbol{x}')$, as a labelling of every point. The definition, eq. (12.4), tells us that, just like the example of a network trained on one data point, the labelling at $\boldsymbol{x}'$ is being generalised to the point $\boldsymbol{x}$ according to the expansion-layer similarity, $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{y}(\boldsymbol{x})^T \boldsymbol{y}(\boldsymbol{x}')$. The output of this operation, eq. (12.4), at a particular point, $\boldsymbol{x}$, corresponds to a weighted sum of the labellings of other points, with the weighting decided by the kernel function between the points, $k(\boldsymbol{x}, \boldsymbol{x}')$. If, for all points where the similarity is large and positive, the labelling, $v(\boldsymbol{x}')$, is also large and positive then they will constructively interfere, producing a large output, and in the case of eigenfunctions, a large eigenvalue, fig. 12.2E green. On the other hand, if the kernel is large and positive across a region in which the label, $v(\boldsymbol{x}')$, oscillates between positive and negative it will produce a correspondingly smaller output, and a small eigenvalue, fig. 12.2E purple.

Matching the intuitions we've developed, the cited theory tells us that the higher the eigenvalue of a given eigenfunction, the fewer training datapoints are needed to generalise the function to a given level of accuracy. Further, a function which is not an eigenfunction can, thanks to the linearity of the problem, be decomposed in the eigenbasis and each component will be learnt independently, with a speed (in terms of training points needed needed to reach a given level of generalisation accuracy) set by the corresponding eigenvalue.

Conclusion These results give us a way to answer our initial question. The connectivity matrix somehow changes the structure of the representation in the expansion-layer. This can be summarised through changes to the kernel function, which measures representational similarity in the expansion-layer. These changes to the kernel cause corresponding changes to the eigenstructure, changing the inductive bias of the network as a whole. We can therefore understand the functional role of different connectivity structures by looking at their effect on the network's eigenstructure.

12.4 An Analytic Link Between Connectivity and Inductive Bias

In this section we model the observed connectivity properties and show their influence on the inductive bias of the network. In general, calculating a kernel's eigenstructure analytically is challenging, so in pursuit of some minimal results we use a very simple connectivity model. We generalise these findings to more realistic settings in the last section.

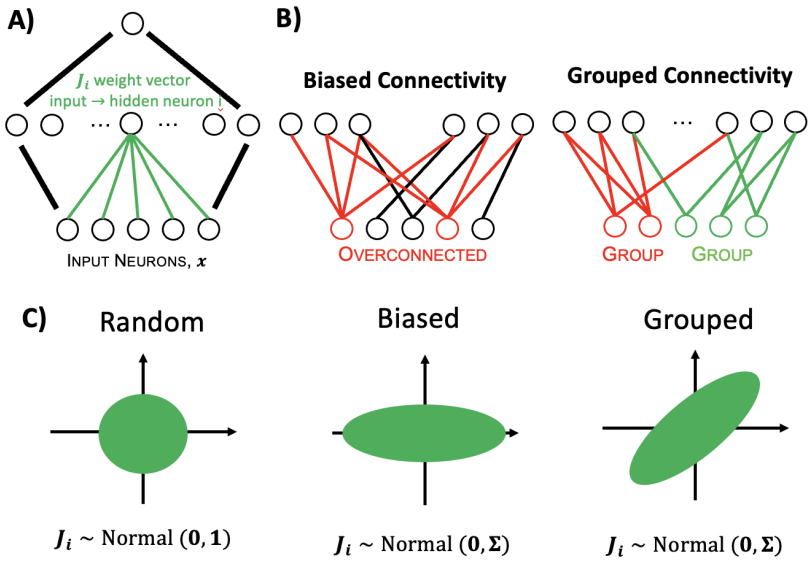


Figure 12.3: **A)** We sample each expansion layer weight vector, $\boldsymbol{J}_i$, independently. **B)** We aim to model two structural motifs - biased connectivity, in which some input neurons connect more than others; and grouped connectivity, in which the inputs are grouped, if a hidden layer neuron is connected to one member of the group it is likely to be connected to others. **C)** We build an analytically tractable model of these structures using simple multivariate Gaussians. Random connectivity is modelled by sampling each weight from a Gaussian with mean zero and variance 1. Biased connectivity is implemented by increasing the variance of one dimension, while grouping is implemented by correlating each hidden neuron's connection to two of the input neurons, defining a group.

Connectivity Schemes Call the vector of input weights to the ith expansion-layer neuron $\boldsymbol{J}_i$, fig. 12.3A. Our model specifies the connectivity matrix through specifying a probability distribution on $\boldsymbol{J}_i$, from which each expansion-layer neuron samples its input weights. We assume there are infinitely many expansion-layer neurons (one could call it 'the infinitely wide nose assumption'), which is not a bad assumption in these cerebellar-networks with their large expansive projection. We then calculate the resulting kernel. To enable tractability, we constrain our the distribution of $\boldsymbol{J}_i$ to be a multivariate Gaussian:

$$\boldsymbol{J}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \tag{12.5}$$

This minimal model allows us to study the effects of the observed connectivity structures through changing the covariance matrix, $\boldsymbol{\Sigma}$. Most simply we have the null model, random connectivity, $\boldsymbol{J}_R$, corresponds to $\boldsymbol{\Sigma} = \mathbb{1}$, the identity matrix, and each weight is sampled independently from all the others.

To study the two observed connectomic effects we introduce two simple changes to $\boldsymbol{\Sigma}$. Empirically, expansion-layer cells connect to some input neurons with higher probability than others (Nguyen et al., 2023; Zheng, F. Li, et al., 2022). We model that by stretching the covariance along one input axis relative to the others, fig. 12.3C. This means the corresponding input neuron will have larger weights, and hence more influence on the representational similarity.

Similarly, there are correlations in the connectivity: the projection neurons form groupings such that if a kenyon cell is connected to one member of a group it is likely to be connected to the others (Zheng, F. Li, et al., 2022). In our simple model we'll group together a pair of the input neurons. To reflect this in connectivity we introduce correlations into $\boldsymbol{\Sigma}$ such that if a kenyon cell is strongly connected to one of the paired projection neurons it is likely to also be strongly connected to the other, fig. 12.3C. Given all these assumptions a simple extension of the well known result of Cho & Saul (Cho and Saul, 2009) allows us to derive the kernel for each connectivity scheme, as in previous work (Pandey et al., 2021). We will use this to understand each connectivity pattern's effect on the inductive bias.

Inductive Bias of Randomly Connected Network For a random connectivity matrix, $\boldsymbol{J}_R$, the kernel depends only on the angle between inputs in the input space, θ, and the length of the vectors (which are all length 1 by assumption), (Cho and Saul, 2009; C. K. Williams, 1998):

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{|\boldsymbol{x}||\boldsymbol{x}'|}{\pi}[\sin(\theta) + (\pi - \theta)\cos(\theta)]. \tag{12.6}$$

The eigenfunctions of this kernel are spherical harmonics (C. Müller, 2012), the extension of sine and cosine to a sphere. The higher the frequency the lower the eigenvalue, fig. 12.4A. Therefore with $\boldsymbol{J}_R$ the network embodies a simple smoothness function.

Effect of Biased Connectivity The changes introduce by the first connectivity motif, stronger connection to some input neurons, can be summarised by the same kernel formula, eq. 12.6, we simply have to change the definition of θ and $\boldsymbol{x}$. Rather than simply representing the datapoints and the angle between them, first stretch your input data by the square root of the covariance matrix, then measure the angle and norm of the stretched datapoints, fig. 12.4B. This change amplifies the effect of variations in the activity of the over-connected input neuron meaning it has a larger effect on the expansion-layer representational similarity. This makes a lot of sense, more connections simply increase its influence! This change manifests intuitively in eigenstructure; eigenfunctions with variation along the overconnected input direction are easier to learn, and hence have higher eigenvalue than those along orthogonal direction, fig. 12.4B.

This leads us to our first intuitive conclusion. Networks that overconnect to some inputs find it easier to learn functions that depend upon variation of that input relative to the others. This matches the claim that monodours that perhaps only activate one projection neuron can still be well identified if they stimulate an overconnected projection neuron, (William Dorrell, Yuffa, and P. Latham, 2023; Zavitz et al., 2021).

Effect of Correlated Connectivity Correlations in the connectivity can be similarly understood. The kernel equation is again identical with only the definitions of $\boldsymbol{x}$ and θ changing. Again, under the changed connectivities they correspond to the angles and sizes of input datapoints after streching by the connectivity covariance, $\boldsymbol{\Sigma}$. In fig. 12.4C the x and y input neurons form an over-connected group. Variations in the activity of all members of this group together, variation along the $x + y$ direction, cause large changes to the expansion-layer representation. Conversely, variations in which some members of the group activate and others quieten, i.e. variation along the $x - y$ direction, have a dampened impact, since most expansion-layer neurons are influenced by the sum of the corresponding changes, so variation along this direction are invisible to the expansion layer representation. The effect on the eigenstructure is again intuitive, fig. 12.4C. The network is inductively biased towards labellings that assign similar valences to datapoints that activate the entire group of projection neurons

in concert, fig. 12.4C. Conversely, labellings that require differentiating between inputs patterns that vary in how they distribute activity within the group while maintaining the same sum are much harder for the network to generalise.

Hence, we reach our second intuitive conclusion. Correlations in the connectivity encourage generalisation across the activity of members of the connected group, making it easy to distinguish odours that tend to activate the whole group, and hard to make distinctions between inputs that activate some members of the group but not others (William Dorrell, Yuffa, and P. Latham, 2023; Zavitz et al., 2021).

12.5 Generalisation to & in More Biological Models

The analytic model presented above suffers from some obvious mismatches with biology. We'll try and correct some of those mismatches now. While these changes mean we cannot analytically calculate the kernel, we can use numerics to derive similar quantities. For low-dimensional inputs it is very easy to densely sample the input space, compute the kernel implied by the network on these points, and then numerically calculate the eigenfunctions. The simplicity of this approach makes it a great first pass to understand how changes to the network impact its learning ability - to encourage adoption of this approach we share all our code, which should be easily adaptable to settings of interest[2].

Biologically Plausible Model In particular we change two things. First, We remove the ReLU nonlinearity and instead, inspired by the large inhibitory Anterior Paired Lateral neuron that reciprocally inhibits the entire kenyon cell population in the fly mushroom body, modify the representation such that only the top 10% of cells are active at a given moment (Lin et al., 2014).

And second, rather than using all-to-all input-to-hidden connectivity and modulating their strengths to encode structural motifs, we, more prosaically, create a sparse connectivity matrix. In the random model we

[2]Code at: `https://github.com/WilburDoz/Infinitely_Wide_Noses`

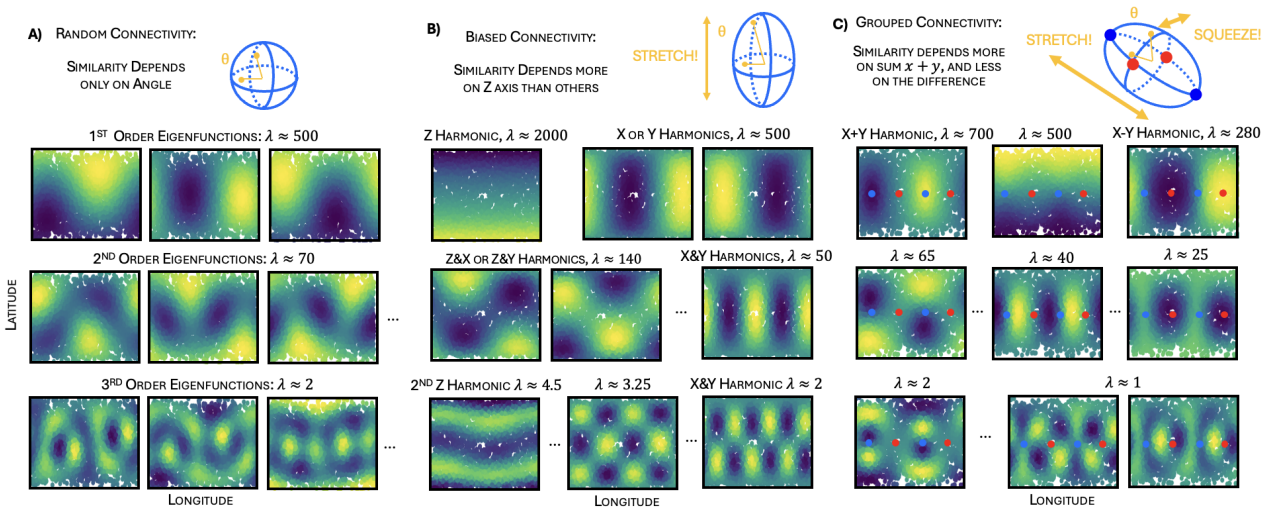


Figure 12.4: **A)** With unbiased random connectivity the similarity depends only on the angle between points on the sphere. The eigenfunctions are assignments of a label to each point on the sphere, we therefore display them projected into 2D space (like a map of the world) with the label denoted by the colour. The eigenfunctions of this kernel are known to be spherical harmonics (C. Müller, 2012). Spherical harmonics fall into groups, there are $2l+1$ members of each group where l is the order of the group, for example the first group contains 3 functions each of which is a smooth oscillation pointing in 3 orthogonal directions. The subsequent orders contain more functions, more oscillations, and have a lower eigenvalue, so are harder to learn. **B)** Biased connectivity can be understood by stretching the sphere along the overconnected axis - which we've chosen to be the z axis. This breaks the degeneracy between spherical harmonics of the same order: functions that oscillate along the z direction have higher eigenvalue and are correspondingly easier to learn than functions that oscillate in the x or y direction. **C)** We implement grouped connectivity by positively correlating the size of the weight vector along the x and y directions. The corresponding kernel can be understood as measuring similarity along an ellipse stretched in the $x+y$ direction (blue dots) while being squeezed in the $x-y$ direction (red dots). This similarly breaks the degeneracy amongst spherical harmonics of the same order: variations along the $x+y$ direction (e.g. top left) have a higher eigenvalue, while variations along the $x-y$ direction have a lower one (top right).

connect pairs of neurons at a fixed probability; we encode overconnection to a given input neuron by increasing its probability of connection; and we introduce the grouping motif by ensuring that if a given expansion layer neuron connects to one member of a group the probability of connecting to others is higher.

In fig. 12.5 we show that the same broad conclusions are true of this more plausible model as in the toy analytic case. In particular, randomly connected networks are biased towards smooth functions; networks overconnected to a given input find it easier to learn functions that rely on the activity of that neuron; and grouped connectivity leads to a bias towards functions depend on the shared activity across members of the group. This network could be easily extended to explore other interesting effects. Simple plausibility fixes could constraining the connectivity weights or the input activities to be positive, while more interesting neuroscience questions could include studying the impact of whitening pre-processing in the olfactory system (Wanner and Friedrich, 2020), or the impact of ongoing neurogenesis in the dentate gyrus (Seri et al., 2001).

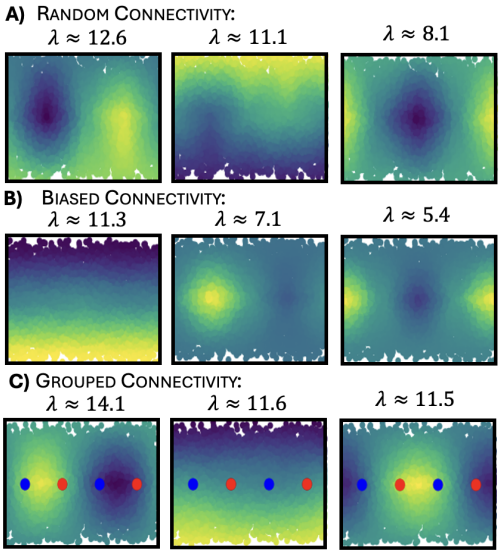


Figure 12.5: Rather than implementing the connectivity biases via correlations in weight values, we show that similar ideas generalise to a network where the biases in connectivity are implemented through connection probabilities, and with a more realistic nonlinearity. Here the connectivity matrix is sparse. The probability of a non-zero connection is either **A)** uniform, **B)** higher for one input neuron, or **C)** high for a particular input-hidden neuron pair if that hidden neuron is already connected to another input neuron in the same group - in this case we again define the group to be the x and y input neurons. In each case show the first three eigenvectors. **A)** Networks are biased towards smooth functions, **B)** but overconnecting in the z direction creates biases towards variations along the z direction. **C)** Grouping the connectivity makes one mode in the grouped direction easier to learn, though interestingly the alignment of this mode changes relative to the previous section - red and blue points as in fig. 12.4.

12.6 Discussion

In this work we have studied the effect of non-random connectomic structure on the inductive bias of cerebellar-like networks. In particular, we have formalised this interaction by describing how toy models of the connectomic structure lead to changes in the learning properties of the network, via their effect on the kernel eigenstructure. In short, as reported in other work, overconnection to some input neurons over others makes it easier for the circuit to learn functions that depend on that input, while groupings in the connectivity make it easy (hard) for the network to learn functions that generalise over (discriminate between) members of the same group (William Dorrell, Yuffa, and P. Latham, 2023; Zavitz et al., 2021).

Broadly, this work aligns with approaches that seek to understand design choices in cerebellar-like networks through the tasks they help to solve (William Dorrell, Yuffa, and P. Latham, 2023; K. D. Harris, 2019; M. Xie et al., 2022; Zavitz et al., 2021), in contrast to task-agnostic approaches that have sought to characterise cerebellar representations via measures such as the dimensionality of the representation (Babadi and Sompolinsky, 2014; Cayco-Gajic, Clopath, and Silver, 2017; Cayco-Gajic and Silver, 2019; Litwin-Kumar, K. D. Harris, et al., 2017). We find this view to be extremely effective, and our main motivation for writing this paper is to didactically illustrate one particular method, kernel regression, that uses this view.

Three different approaches have been used to understand the inductive bias of cerebellar-like networks, in particular the impact of structured connectomic patterns. Each method has its own merits and drawbacks.

Zavitz et al., 2021 demonstrate the first approach: choosing a battery of tasks and testing a network on these tasks with or without the key design feature. This is simple, and the network models can be arbitrarily complex, but requires the user to guess a reasonable battery of tasks, which is not always easy. William Dorrell, Yuffa, and P. Latham, 2023 instead meta-learn tasks which the network finds most easy. At the cost of methodological complexity, this retains the arbitrary complexity of the model while obviating the need for guessing reasonable tasks, instead the method searches in the space of all tasks for those that the network finds easiest to learn. Finally, in this work we have highlighted the potential of using kernel regression, as in previous works (K. D. Harris, 2019; M. Xie et al., 2022). This approach constrains learning to a set of linear output synapses, and requires further constraints to allow analytic derivations of the kernel or its eigenfunctions, however it offers the most precise and complete understanding. Not only does it share the meta-learner's ability to tell you which, among all tasks, the network will perform well at; it links to a flourishing, and still developing, theoretical basis, letting you do principled things like decompose a task into parts - the eigenbasis - and understand how each of these will be learnt.

One shared shortcoming of these approaches is the difficulty of interpretation for high input dimensions. When the inputs are low-dimensional, simple pictures, like fig. 12.4, suffice to understand a network's inductive bias. On the other hand, for high-dimensional inputs all three methods struggle: it becomes hard to guess reasonable task batteries, to meta-learning function, or to numerically calculating the eigenfunctions. It is in this setting in particular that analytic approaches hold promise, since equations trivially extend beyond three dimensions. Tractable toy models can be established, as here, that demonstrate the key phenomena while generalising to higher input dimensionalities. This can require contorting the problem into a solvable form while capturing the key biological phenomena, which is not always possible. That said, we are hopeful that recent developments in machine learning theory could usefully loosen the requirements for analytically tractability, providing dividends for understanding cerebellar-like networks (H. Hu and Lu, 2022).

The normative approaches we are advocating are appealing, but have a whiff of untestable pseudoscience about them. It seems a shame to declare victory when a normative theory 'explains' a phenomena without making testable experimental predictions. Fortunately, this is not impossible. In some cases, such as the impact of representational sparsity, the variable of interest could be perturbed online, for example by inactivating the inhibitory neuron that enforces the sparsity of the representation in the drosophila mushroom body. Crazily, even more developmentally hard-wired and inaccessible variables, such as the number of neurons in the kenyon cell layer of the mushroom body, can be varied with genetic tools (Ahmed et al., 2023), and the resulting engineered animals could then be tested on a battery of learning tasks. Finally, evolution often provides natural experiments in which parameters of interest vary across a related group of animals (K. E. Ellis et al., 2023).

To conclude, this work provides a didactic presentation of using recent results on kernel regression to understand the effect of connectivity on the inductive bias of cerebellar-like networks. It is hoped that this kind of approach can serve as an easy first-pass analysis to understand the effect of circuit features on the inductive bias of cerebellar-like networks. Between this, and other work (William Dorrell, Yuffa, and P. Latham, 2023), we hope to lower the barrier to using these approaches to understand cerebellar-like networks.

Part V

Conclusion

Chapter 13

Conclusion

13.1 Overview – Major Theme

Neuroscience attempts to establish how intelligence emerges from its biological substrate. The version of this question found in this thesis asks 'what is the mapping between neural firing rates and the algorithm they implement?' In answering this question, neuroscientists have used many streams of evidence; in this thesis we developed one type - normative models of neural firing (and minorly, of connectomic structure). These are optimisation problems that study the optimal structuring of neural firing to achieve various imagined goals.

In the introduction we outlined the two previous approaches that exist for constructing such theories. First, the paradigmatic example is the efficient coding hypothesis. These models optimise neural firing to maximally encode information subject to various efficiency constraints. Indeed, part II of this thesis explicitly studied these types of models. However, they fail as models of a lot of neural activity because neurons are not just communicating, they are computing with the variables they encode, and neural activity contains traces of this computation. In chapter 1 and chapter 4 we outlined how grid cells are a good counterexample to the efficient coding hypothesis when applied to recurrently computing neural circuits, like cortex. Second, we discussed connectionism, using task-optimised artificial neural networks as models of neural activity. We saw how these models escape the failings of efficient coding, allowing them to model computing neurons, such as grid cells. However, they are inscrutable, leaving us a similar, though thankfully slightly easier, question: 'why does neural activity, in both the brain and our artificial neural network models of it, structure itself in this way, not that?'

We therefore constructed a tractable normative theory of neural firing implementing meaningful computations, which we've been calling the efficient computing hypothesis. This can be understood from one direction as adding a computability constraint to an efficient coding optimisation problem to ensure the code can implement the relevant algorithm. Alternatively, we could frame it as studying the optimal configuration of simple connectionist models: gated linear recurrent neural networks. Via this simplification, we were able to create understandable normative theories of neural activity engaged in two computations: the grid cell representation of path-integration – part I, and the prefrontal slot representation of structured working memory – part III.

We hope that, with further work, variations and elaborations on these approaches will produce a useful, interpretable, theoretical toolbox for linking neural firing to algorithms, permitting the use of neural firing to infer the brain's algorithms, and the use of behavioural evidence to understand puzzling neural patterns.

In the following sections we give a brief overview of the other parts of this thesis, before then discussing other work from my PhD that I chose not to include. Then, we outline limitations and future work before closing with a brief broader discussion.

13.2 Overview - Minor Themes

The main argument presented above misses two parts of this thesis, which we review here for completeness.

First, in part II, we asked when neural circuits and representations were likely to be modular, breaking apart the computation into meaningful subparts implemented by disjoint sets of neurons. This is pivotal, since only through this modularity do we have a hope of understanding neural circuitry. To study this question, we reverted to a simple efficient coding model and derived the conditions under which such a model would split the encoding of different variables into disjoint neurons, chapter 5. We then partially extended these results to a recurrent, 'efficient computing' setting, chapter 6, which enabled us to understand both the modularisation of grid modules, as well as patterns of mixed-selectivity and modularity observed in grid cells reward warping, chapter 7. Finally, we framed a large family of efficient coding problems that, with the right change of variable, could be seen as convex optimisations, chapter 8. We used this result to identify both matrix factorisation problems and neural tuning curves, before studying ON and OFF channels in the retina. Overall, this part of the thesis moved to

efficient coding, rather than efficient computing, for analytic tractability. We hope that extensions will make this move unnecessary, as briefly developed in chapter 6.

In most of this thesis we studied neural circuits implementing a fixed input-output function. However, one of the most impressive abilities in the brain is learning. In part IV, we briefly thought about similar normative approaches to one of the cleanest learning circuits in the brain – cerebellums and other cerebellar-like structures. We used both meta-learning approaches, chapter 11, and recent results from machine learning theory, chapter 12, to understand how measured connectomic patterns in cerebellar-like circuits can be normatively interpreted through the way in which they make some, presumably pertinent, associations easier to learn than others.

13.3 Overview – Missing Themes

There are a few parts of my PhD work that are not included in this thesis because either I did not lead the work or it was substantially incomplete. Three perhaps warrant brief mention: a striatal replay detection project, two applications of this work in modern machine learning, and a variety of attempts to verify our neural predictions experimentally.

13.3.1 Point Process Models for Detecting Striatal Replay

First, and most unrelated to this thesis, I worked with a few collaborators to modify an existing point process model of sequential spiking data to detect replay in an unsupervised fashion (the, relatively useable, code is available at https://github.com/lindermanlab/PPSeq.jl/pull/15). We used this to find evidence for replay of motor sequences in dorsolateral striatum (Thompson et al., 2024), and our collaborators have used these techniques to show that this replay occurs independently of hippocampus, the first time this property of replay has been shown.

13.3.2 Links to Machine Learning

Second, as discussed, our normative representational theories are purposefully very close to more standard connectionist models, like feedforward and recurrent neural networks. This leads to some natural cross-pollination with machine learning; indeed, I've been heavily influence both by connectionist work, and machine learning theory. In the other direction, twice in my PhD I have used our theories for purely machine learning ends, both related to the work presented in part II.

Identifiability of Sparse Autoencoders The first of these concerns sparse autoencoders, dictionary learning/sparse coding models like those used to model edge detectors in V1 (Olshausen and Field, 1996). Recent work has used these to study the internal representations of large language models (LLM), i.e. the input data to be sparsely encoded then reconstructed is the neural activity in a hidden layer of one of these large networks. They have found that the elements extracted by dictionary learning are meaningful, encoding concepts like 'the golden gate bridge' (Bricken et al., 2023). Further, astoundingly, including these sparse expansions in the forward pass of the LLM and then artificially activating a concept cell, like the golden gate bridge neuron, leads the network to insist on bringing up the golden gate bridge in conversation – i.e. these extracted concepts seem to be more than mere hallucinations, they are causal. I have used similar results to the matrix factorisation work presented in chapter 8 to derive identifiability results for sparse autoencoders, outlining exactly which concepts a sparse autoencoder will extract. This has been useful in understanding some puzzling phenomenology such as 'feature splitting' (Templeton et al., 2024).

Developing State-of-the-art Disentangling The second set of related work is in disentangled representation learning. This is a subfield of machine learning that seeks to learn the meaningful subparts of arbitrary datasets. For example, the meaningful subparts of a set of portraits might be things like hair colour, orientation, and the presence of glasses. I spent a joyful three months rotating in Stanford University, where I worked with (now Dr.) Kyle Hsu. We studied the disentangling properties of the efficient coding problems presented in part II. I then played a minor role helping Dr Hsu develop a related state-of-the-art disentangling method, QLAE, which maps images to a discrete grid of points in a latent space (K. Hsu, William Dorrell, et al., 2023; K. Hsu, Hamid, et al., 2024). It turns out that many of the theoretical results presented in part II, such as the critical role of range independence, are also seen in QLAE, suggesting a deeper correspondence between frontier machine learning approaches and our relatively toy theoretical settings that could usefully be further explored.

13.3.3 Experimental Tests of Neural Predictions

Finally, in this thesis we have presented three different theories of neural firing: a theory of grid cells, part I; a theory of mixed-selectivity and modularity, part II; and a theory of structured working memory slots, part III. We may have succeeded in postdicting some data, but the real test is prediction. Experimental tests of each of these ideas are currently under development in the Behrens' lab. Of these, I've been most involved in designing and running experiments to test the grid cell theory.

Grid Cell Test Given the difficulty of measuring many grid modules simultaneously or estimating the number of grid cells in a module, we settled on testing the predictions regarding the adaption of grid modules to differently shaped rooms using electrophysiology (neuropixels) in mice, chapter 3.

Our experimental proposal begins by measuring the lattices of two modules of grid cells in a large room. Our theory prescribes which single module lengthscales are optimal in each room geometry, fig. 3.5; hence, each of the two measured modules will be optimal for a certain set of room sizes. We'll therefore adapt the room size to match an optimal room for one of the previously measured grid modules is optimal, but not the other. The prediction would be that the grid module that is optimal in the new room shouldn't change its lengthscale, while the other should, as observed by H. Stensola et al. (2012) and modelled in chapter 3. Doing this concurrently with two modules removes confounds while also cleaning up some of the indeterminacy of the theory: while the theory predicts clean optimal single modules, when there are multiple modules it becomes very hard to predict their lengthscales. This experiment therefore relies on a simpler prediction: if a grid cell is already optimal for a given room, it probably shouldn't change its firing pattern. Even this is likely not always true theoretically, but it is more robust. Further, this experimental design is appealing even if we can't measure two modules in one animal, as we could still test whether we can control the lengthscales of single module by changing the room size.

We ran this experiment in Spring 2025, but sadly it appears that none of the electrodes found a grid cell, likely because the probes missed the entorhinal cortex. We ran one pilot animal in Autumn 2024 in which we did measure grid cells. Unfortunately, however, the probe broke before we could run the experiment described above.

We did, however, get one week of grid cell measurements, during which we were debugging the data analysis pipeline. During this time we ran a different experiment, whose motivation and construction I'll now briefly describe. It is not well understood how grid cells encode curvy spaces. Such spaces, for example a hilly landscape, are interesting because they require the path-integrator to adapt its integration to the gradient. Previous work on grid cells on curvy 2D surfaces has only measured on 2 differently oriented 2D planes (Hayman et al., 2015), which is not sufficient to test whether grid cells can perform such curvy path-integration, because two 2D planes can be isometrically embedded into 2D space. This is not true in general, for example, I constructed two steeply sloped square-based pyramids (45 or 60 degree slopes), fig. 13.1, and a large hemispheric surface, neither of which can be isometrically embedded into 2D space. We measured six grid cells from the pilot animal on the two pyramids and found warped grid-like coding, which future analysis could usefully extensively explore, fig. 13.1.

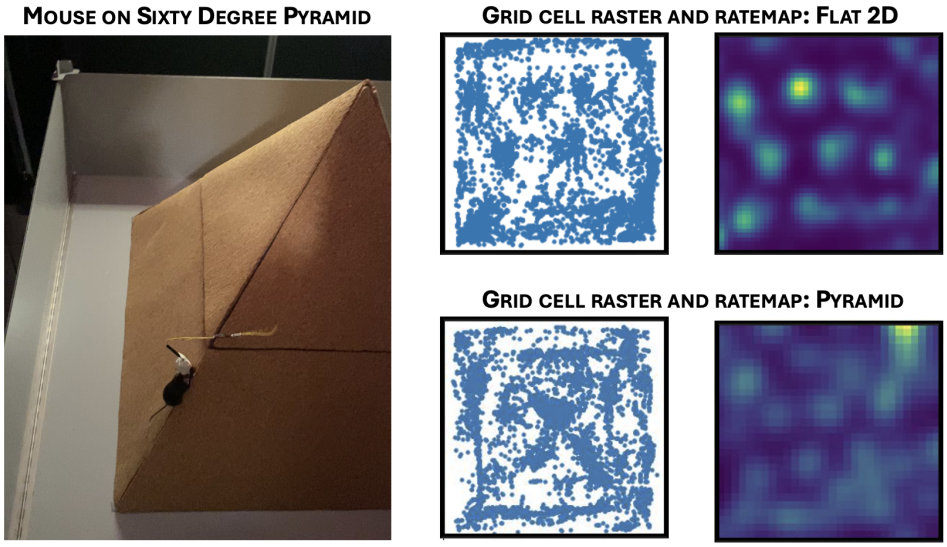


Figure 13.1: Mouse climbing on a square based pyramid with slopes at sixty degrees. The ratemaps of a grid cell are shown as a function of the xy position of the mouse either with or without the pyramid. The cell is clearly a grid cell, but the firing fields morph in the presence of the pyramid. The outline of the pyramid can be seen in the raster, since the mouse spent more time running around the base of both the upper and lower segments of the pyramid and along the corners.

Modularity & Mixed Selectivity Test part II explains how, in our theories, the key determinant of modularisation is range independence, and this idea explains observed experimental differences in whether grid cells warp to reward. In brief, it depends on the distribution of rewards, if space and reward are range-dependent the grids warp, becoming mixed-selective Boccara et al. (2019); alternatively, if space and reward are range-independent the grid cells should remain pure (Butler, Hardcastle, and Giocomo, 2019). This is a very testable prediction, we should be able to run two experiments that differ only in the distribution of the rewards and see different warping effects. Arya Bhomick is hoping to run these experiments with help from Diksha Gupta, though this experiment remains in the planning stage.

Prefrontal Working Memory Test part III describes our theory of structured working memory representations. A key postdiction we made was, as above, that the (range-)independence of different memories was a key driver in whether their encoding subspaces were aligned or orthogonal. Diksha Gupta is training rats on sequences of nose pokes and hoping to measure exactly these kinds of effects – the rats have been trained, and we await neural recordings!

In sum, significant effort has been invested in testing these theories, though, as yet, we have no results. These seem vital, both to check whether the theories are valid, and to tune the theories. For example, choosing to penalise either the L1 or the L2 norm of neural firing produces slightly different patterns of modularisation that could be experimentally distinguished.

13.4 Limitations & Future Directions

This thesis presents limited progress towards the grand goal of the introduction: a normative theory for neurons performing computation. There are numerous limitations with our approach; we'll briefly run through a few and use them to describe some of my planned future work.

13.4.1 Limited Validation → Proposed Tests

First, while we tried repeatedly to match out theoretical predictions to data, with some successes, the evidence is still weak. This motivated the experiments designed above, and running these and other reanalyses of existing data are promising directions. Further, even in the datasets we did compare to, there were important aspects of the neural data we were unable to explain. In particular, the encoding of task-progress (aka phase), and the single-neuron predictions of mixed-selectivity vs. modularity in lateral prefrontal cortex differed from our theoretical predictions, part III. Optimistically, these two mismatches seem to be linked, reflecting the simplicity of the computation we are assuming is performed by these circuits. It seems that thinking more deeply about the computation these conjunctive phase-'lagged position' encodings permit could lead both to better normative theories, but more importantly, to a better understanding of the underlying computation. In this, normative theorising shows its appealing role as null hypothesis: if the circuit really was just performing structured working memory as we formalised, the representation would make more sense. Since it doesn't, there must be something else going on. Whether this is true, or just an overestimating theoretical fever-dream remains to be explored.

13.4.2 Limited Scope → Other Applications

Second, while the introduction promised a normative framework for generic cortical/recurrent computation, the reality was two 'map-based' neural circuits, suggesting the scope of our theory is much more limited. This, however, seems to reflect time limitations rather than anything fundamental. I've had some preliminary success thinking about both orbitofrontal value encodings, and supplementary motor area representations using these ideas. In many ways, this should not be surprising. Recurrent neural networks (RNNs) are a pervasive model of most cortical areas, and our theory is simply a more analytically tractable approach to such circuits. Since RNNs succeed in motor cortex (Schimel, Kao, and Hennequin, 2024), orbitfrontal cortex (Z. Zhang et al., 2018), parietal (Pouget and Sejnowski, 1997), or sensory (Todorov, Siapas, and Somers, 1996), cortices, we should expect analytic analysis of such circuits to be useful there too. Further, classic ideas of a cortical column suggest that the computation performed across cortex is shared. This attractive idea would suggest that a theory that performs well in one cortical area could, with some twists, work elsewhere. We look forward to testing whether any of this is true.

That said, there are many neural components that simply do not fit naturally into the current framework. The brain has complex ongoing plasticity, dopaminergic neurons that whose extensive projections modify this plasticity as well as other broad neuromodulators, hippocampal hebbian learning, complex replay mechanisms, bursting and oscillations. Our theory will not explain these phenomena, among many others. That is what it is; our theory is a theory of neural firing implementing fixed computations. This is already a challenging problem whose relevance to the brain make it useful to puzzle over. Further integration, for example with

ongoing plasticity or replay that are thought to be important in some of the brain regions we have discussed (Compte et al., 2000; Schwartenbeck et al., 2023), will have to wait for another thesis.

13.4.3 Computationally Limited → Theoretical Extensions

One limitation I have often felt is my inability to match interesting computations to satisfying normative frameworks. We discussed a few interesting computations in this thesis, but the theoretical tools are currently very brittle to even relatively simple changes. Thankfully, there seem to be many fruitful directions for generalising, which we'll now discuss. These fall into two broad categories: structural generalisation, and more realistic circuit implementations.

Structural Generalisations First, some of our cleanest analysis relied on the group structure of the problems, chapter 2. If the set of 'actionable matrices', the matrices that perform the computations, form a representation of a group, then group representation theory provides a lot of tools for understanding the structure of the resulting neural code. However, groups are very crystalline, requiring constricting axioms such as the existence of inverses: 'for every action you take there is another that takes you back'. Yet we grasp and manipulate many looser forms of structure that break this and other rules, such as the intricate substructures of even simple board games. Thankfully, the representation theory of finite semi-groups, a much more relaxed algebraic structure, is relatively well-developed, and the first textbook on the subject was recently published (Steinberg et al., 2016). Further, semi-group structure is analogous to a finite state-machine, suggesting that a generalised theory that uses semi-groups rather than groups could be used to understand the implementation of any finite state machine.

Another direction of structural diversification is suggested by our curvy two-dimensional pyramid grid cell experiments. The set of rules exhibited by navigation on a pyramid could be encapsulated by a finite state machine, but in this case, there is a much more natural description. The problem remains navigation in two dimensions but with a modulation dependent on which surface of the pyramid you're currently on. There is an area of maths that appears constructed for exactly these kinds of situations: sheaf theory, in particular, cellular sheaves (Hansen and Ghrist, 2019). To the best of my current understanding, this concerns encoding relations between nodes on a graph using linear algebra. The application here would be encoding the relationship between different actions on the different surfaces of the pyramid using these sheaves. I would be excited to see whether such a theory could explain our grid cell recordings from curvy 2D. Further generalisations of these ideas could apply to any case of adapting structure to a changing context, such as performing Bayesian updates of sensory variables adapted by the error bars on the variable.

These thoughts point to a deeper problem: we presented two alternative structural diversifications – arbitrary finite state machines, or a set of relations between actions on different sides of a pyramid. How are these related and how might one know which to use? I think this question can be best understood as a question about hierarchical structure. The finite state machine framework discussed above is hierarchically flat, with two simple layers: a set of states and a set of actions to move you around them. The cellular sheaf framework, however, has depth: actions, the 2D movements, exists on all surfaces of the pyramid, but their meaning is modulated by a higher-level encoding of which surface of the pyramid you are on. This is natural. We clearly exploit hierarchical structuring regularly – a sequence of digits 080808080808 is not remembered as 12 digits but as six repeats of 08 and our neural circuits will reflect this. I have been dreaming of a theory that will describe which hierarchical decomposition is optimal in each problem for a long time, but have yet to find the right wording. Empirically, training RNNs is clearly an option, and related work has been done in the language of thought area (K. Ellis et al., 2020). I think a mixture of the ideas presented here will be fruitful in understanding how the RNN chooses to hierarchically decompose the task. but this remains speculative.

More Realistic Circuitry Our model of neural computation is clearly quite abstract. We model circuitry that is well described by continuous attractor circuits, yet, without noise, we have no concept of an attracting region: the activity lives on the ideal manifold and never deviates. It is a choice whether to model these effects, and personally I don't think including such effects in these models is a productive use of time. There are already well-established models that explain how centre-surround connectivity or local excitation and global inhibition can structure an attractor circuit. If we can understand the relationship between neural firing and computation to reasonable accuracy while abstracting this process, just like we abstracted neurons' temporal code or vesicle release probability, then that is to be celebrated.

However, there is one aspect of realism that I would like to incorporate in future: a more realistic gating mechanism. In this thesis, actions controlled which recurrent weight matrix was used. However, as discussed in chapter 2, in reality, 'shifter neurons' that conjunctively code state and action perform this effective gating. By abstracting away this process we forfeit predictive power over these neurons, hence why we were unable to

model the conjunctive position-heading neurons in the grid system part I. I find this overly limiting for two reasons.

First, these neurons seem key to answering a very interesting cognitive question: 'what is an action?' When navigating in the space of rotations it might seem natural to break actions into leftward and rightward turns. But on a more generic structure, like the space of chess boards, the relevant notions of actions might not be so obvious. For example, perhaps a good choice of action is a high level structural relation: 'a more offensive right flank'. The simplest version of this question seems to be asking how, optimally, a neural circuit should choose to structure its shifter neurons. Perhaps, given the structure of chess and the distribution of natural chess games, the most efficient set of shifter neurons to learn might include more abstract concepts, beyond just 'move knight forward and right'.

Second, these neurons are a precise form of neurally implemented hierarchy. Shifter neurons are often adaptive, a process that requires a minimal amount of hierarchical depth in the computation. For example, monkeys were trained to listen to a beat – a sequence of clicks at a specific frequency – then were trained to remember the frequency, and repeat it after a delay with their saccades (Merchant and Averbeck, 2017; Merchant and Lafuente, 2024). Neural activity during these tasks was found to follow a stereotyped path, while the speed of movement along the path varied according to the frequency of the beat. This is a three-layer hierarchy with the frequency on the top, the shifter neurons in the middle, and the state neurons (how far through each beat cycle) on the bottom. This is similar to animals adapting their path-integration on inclined slopes, as discussed above. It seems that, towards the goal of understanding the hierarchical decomposition of tasks, a normative understanding of the choices shifter neurons make would therefore be incredibly useful, since it is through them that we see the simplest instantiation of hierarchy.

Given these motivations, I have been dreaming up various normative theories to capture the behaviour of shifter neurons. The simplest incorporate shifter neurons into the standard group representation framing, allowing results from group representation theory to guide reasoning about their activity also, though this formalisation still feels slightly contrived.

Structural Identifiability Broadly, these directions hope to capture how structure and computation are implemented in recurrent neural circuits. This could lead to neural predictions, as discussed and in the vein of the rest of this thesis, including for measured neural activity that is poorly understood, like neurons that perform mental swap operations (Ohbayashi, Ohki, and Miyashita, 2003; Tian et al., 2024), or neural tuning to speed (Betancourt et al., 2023; Bi and C. Zhou, 2020; Crowe et al., 2014; Merchant, Harrington, and Meck, 2013; Merchant, Mendoza, et al., 2023; Merchant, Pérez, et al., 2013; S. Zhou, Masmanidis, and Buonomano, 2022). However, they could also produce purely behavioural predictions, more in the style of classic connectionism (Rumelhart and McClelland, 1985). For example, prediction of which hierarchical decomposition is optimal could be tested by training people on a task and seeing how they think about it, either through verbal report, or by analysing error patterns – an approach that could then be generalised to non-human animals. Finally, these results could be interesting from a purely machine learning perspective. Structural identifiability results could describe what must be true about a dataset such that a particular hierarchical decomposition is learnt, explaining otherwise puzzling RNN behaviour.

13.5 Discussion

This thesis has sought to understand cortical neural activity using tractable normative theories of neural computation. It builds on two existing bulwarks that have stood for over half a century: efficient coding, and connectionism. This slightly old-school approach stands in contrast to recent arguments that, largely motivated by deep learning, suggest these approaches are unlikely to succeed. For example, I listened to a summer school lecture in which a prominent scientist laid out their vision for the future of neuroscience, which comprised a list of the losses and architectures required to produce a deep learning model of each brain area, similar to Richards et al. (2019). It was argued that more first-principles understanding of why artificial neural networks do what they do would require more advanced mathematics than we have access to, suggesting it cannot currently be reached.

Having completed this thesis, the reader will hopefully find themselves broadly agreeing that a loss and architecture, when minimally specified and probed, can form a good model of neural data – the basic connectionist tenet. However, they will hopefully also agree that hope is not lost for a deeper understanding of why our connectionist models fit neural data. Indeed, they might even agree that such an understanding is necessary for us to fully trust our models, since otherwise their complexity ensures they can be easily misinterpreted (Banino et al., 2018). Having spent 5 years on this approach, it is currently my view that, when we understand the algorithm that is being implemented by the neural network, this kind of mathematical description is indeed possible, though not easy.

This points to a large current failure mode: we lack mechanistic understanding of the relevant algorithms. For example, despite decades of study and the recent deep learning revolution, the precise sequence of understandable steps required to turn an image into an object categorisation is not known. It is surely this lack of knowledge that stops us from understanding visual cortex, or the convolutional neural networks (CNNs) modelled on it. We hope that further work theoretically and experimentally probing both visual cortex and CNNs, in their external behaviour and internal representations, will be able to close this gap, and analogous ones in other domains. Further, we hope that normative theories will be able to provide useful null hypothesis and tests, making statements like 'if the circuit was implementing this computation we should see this pattern of activity. We don't, so something else must be going on'.

In pursuing this kind of normative approach, some interesting points emerge, one of which we will briefly note. Looking at single neuron tuning curves can be incredibly informative. This is motivated by our experience with grid cell modelling part I. The precision of the grid cell code and our measurements thereof ensure that only a small number of models can capture even the highest-level features, and all of them use the same set of ideas to do this chapter 4. This convergence is a ringing endorsement for standard, plodding, neuroscientific discovery that pairs intricate neural measurements with minimal reasonable models. Single neurons are clearly meaningful; action potentials are a huge energy expense and biology is heavily incentivised to minimise their use. This ensures that the neuron basis is a meaningful basis in which to study neural activity, and we have found that forcing ourselves to think about single neuron behaviour has been a satisfying whetstone for sharpening our thinking. This stands in contrast to views which place front-and-centre the explanatory utility of population modes independent of their neural instantiation. These approaches can struggle where single neurons succeed, for example, a place cell code and a grid cell code both look like space under a 2D PCA projection, despite their very different computational role. This example presents a straw man view of population approaches, more elaborate approaches would be able to distinguish a grid vs. place code. In general these approaches are clearly useful; indeed, we spent much of part III discussing population subspace codes for structure working memory. Yet, it is largely because we haven't managed to fully describe the single neuron properties of the structured working memory code that I am persuaded there is more to understand, reinforcing the value of single neurons thinking.

In sum, these approaches seem to suggest that the Marr-ian roadmap for systems neuroscience is solid. I remain hopeful that a convergence of all the various streams of evidence available, from single neuron responses and ion channel densities to deep learning models and psychological findings, will lead us eventually to precise understanding of the brain's algorithms and how they are implemented in neural circuits – just as is being established for path-integrating circuits. I hope that the methods described in this thesis will be a useful contribution to this endeavour. This progress can then deliver on the promise of neuroscience: to understand nervous systems and intelligence, helping us to cure its illnesses, control its artificial analogues, and more deeply understand the world around and within us.

Bibliography

Aceituno, Pau Vilimelis, Dominic Dall'Osto, and Ioannis Pisokas (2024). "Theoretical principles explain the structure of the insect head direction circuit". In: *Elife* 13, e91533.

Aharon, Michal, Michael Elad, and Alfred M Bruckstein (2006). "On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them". In: *Linear algebra and its applications* 416.1, pp. 48–67.

Ahmed, Maria et al. (2023). "Hacking brain development to test models of sensory coding". In: *bioRxiv*, pp. 2023–01.

Albus, James S (1971). "A theory of cerebellar function". In: *Mathematical biosciences* 10.1-2, pp. 25–61.

Anderson, Charles H and David C Van Essen (1987). "Shifter circuits: a computational strategy for dynamic aspects of visual processing." In: *Proceedings of the National Academy of Sciences* 84.17, pp. 6297–6301.

Aso, Yoshinori et al. (2014). "The neuronal architecture of the mushroom body provides a logic for associative learning". In: *elife* 3, e04577.

Atick, Joseph J and A Norman Redlich (1990). "Towards a theory of early visual processing". In: *Neural computation* 2.3, pp. 308–320.

— (1992). "What does the retina know about natural scenes?" In: *Neural computation* 4.2, pp. 196–210.

Attneave, Fred (1954). "Some informational aspects of visual perception." In: *Psychological review* 61.3, p. 183.

Babadi, Baktash and Haim Sompolinsky (2014). "Sparseness and expansion in sensory representations". In: *Neuron* 83.5, pp. 1213–1226.

Bach, Francis (2017). "Breaking the curse of dimensionality with convex neural networks". In: *Journal of Machine Learning Research* 18.19, pp. 1–53.

Badura, Aleksandra and Chris I De Zeeuw (2017). "Cerebellar granule cells: dense, rich and evolving representations". In: *Current Biology* 27.11, R415–R418.

Bahri, Yasaman et al. (2021). "Explaining neural scaling laws". In: *arXiv preprint arXiv:2102.06701*.

Baldassano, Christopher, Uri Hasson, and Kenneth A Norman (2018). "Representation of real-world event schemas during narrative perception". In: *Journal of Neuroscience* 38.45, pp. 9689–9699.

Banino, Andrea et al. (2018). "Vector-based navigation using grid-like representations in artificial agents". In: *Nature* 557.7705, pp. 429–433.

Barlow, Horace B et al. (1961). "Possible principles underlying the transformation of sensory messages". In: *Sensory communication* 1.01.

Barry, Caswell et al. (2012). "Grid cell firing patterns signal environmental novelty by expansion". In: *Proceedings of the National Academy of Sciences* 109.43, pp. 17687–17692.

Basu, Raunak et al. (2021). "The orbitofrontal cortex maps future navigational goals". In: *Nature* 599.7885, pp. 449–452.

Bau, David et al. (2020). "Understanding the role of individual units in a deep neural network". In: *Proceedings of the National Academy of Sciences* 117.48, pp. 30071–30078.

Beetz, M Jerome et al. (2022). "Flight-induced compass representation in the monarch butterfly heading network". In: *Current Biology* 32.2, pp. 338–349.

Bein, Oded and Yael Niv (2025). "Schemas, reinforcement learning and the medial prefrontal cortex". In: *Nature Reviews Neuroscience* 26.3, pp. 141–157.

Bell, Curtis C (1981). "An efference copy which is modified by reafferent input". In: *Science* 214.4519, pp. 450–453.

Bell, Curtis C, Angel Caputi, and Kirsty Grant (1997). "Physiology and plasticity of morphologically identified cells in the mormyrid electrosensory lobe". In: *Journal of Neuroscience* 17.16, pp. 6409–6423.

Bell, Curtis C, Angel Caputi, Kirsty Grant, and Jacques Serrier (1993). "Storage of a sensory pattern by anti-Hebbian synaptic plasticity in an electric fish." In: *Proceedings of the National Academy of Sciences* 90.10, pp. 4650–4654.

Ben-Yishai, Rani, R Lev Bar-Or, and Haim Sompolinsky (1995). "Theory of orientation tuning in visual cortex." In: *Proceedings of the National Academy of Sciences* 92.9, pp. 3844–3848.

Bengio, Yoshua et al. (2005). "Convex neural networks". In: *Advances in neural information processing systems* 18.

Berman, Abraham and Naomi Shaked-Monderer (2003). *Completely positive matrices*. World Scientific.

Betancourt, Abraham et al. (2023). "Amodal population clock in the primate medial premotor system for rhythmic tapping". In: *Cell Reports* 42.10.

Bi, Zedong and Changsong Zhou (2020). "Understanding the computation of time using neural network models". In: *Proceedings of the National Academy of Sciences* 117.19, pp. 10530–10540.

Boccara, Charlotte N et al. (2019). "The entorhinal cognitive map is attracted to goals". In: *Science* 363.6434, pp. 1443–1447.

Bordelon, Blake, Abdulkadir Canatar, and Cengiz Pehlevan (2020). "Spectrum dependent learning curves in kernel regression and wide neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 1024–1034.

Bordelon, Blake and Cengiz Pehlevan (2022). "Population codes enable learning from few examples by shaping inductive bias". In: *Elife* 11, e78606.

Borzello, Mia et al. (2023). "Assessments of dentate gyrus function: discoveries and debates". In: *Nature Reviews Neuroscience* 24.8, pp. 502–517.

Botvinick, Matthew M and David C Plaut (2006). "Short-term memory for serial order: a recurrent neural network model." In: *Psychological review* 113.2, p. 201.

Boyd, Stephen and Lieven Vandenberghe (2004). "Convex optimization". In: *Cambridge UP*.

Braun, Lukas, Erin Grant, and Andrew M Saxe (2025). "Not all solutions are created equal: An analytical dissociation of functional and representational similarity in deep linear neural networks". In: *Forty-second International Conference on Machine Learning*.

Bricken, Trenton et al. (2023). "Towards monosemanticity: Decomposing language models with dictionary learning". In: *Transformer Circuits Thread* 2.

Bronstein, Michael M et al. (2021). "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478*.

Buchholz, Simon, Michel Besserve, and Bernhard Schölkopf (2022). "Function Classes for Identifiable Nonlinear Independent Component Analysis". In: *arXiv preprint arXiv:2208.06406*.

Burak, Yoram and Ila R Fiete (2009). "Accurate path integration in continuous attractor network models of grid cells". In: *PLoS computational biology* 5.2, e1000291.

Burgess, Neil (2008). "Grid cells and theta as oscillatory interference: theory and predictions". In: *Hippocampus* 18.12, pp. 1157–1174.

Burgess, Neil, Caswell Barry, and John O'keefe (2007). "An oscillatory interference model of grid cell firing". In: *Hippocampus* 17.9, pp. 801–812.

Butler, William N, Kiah Hardcastle, and Lisa M Giocomo (2019). "Remembered reward locations restructure entorhinal spatial maps". In: *Science* 363.6434, pp. 1447–1452.

Canatar, Abdulkadir, Blake Bordelon, and Cengiz Pehlevan (2021). "Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks". In: *Nature communications* 12.1, pp. 1–12.

Carpenter, Francis et al. (2015). "Grid cells form a global representation of connected environments". In: *Current Biology* 25.9, pp. 1176–1182.

Cayco-Gajic, N Alex, Claudia Clopath, and R Angus Silver (2017). "Sparse synaptic connectivity is required for decorrelation and pattern separation in feedforward networks". In: *Nature communications* 8.1, p. 1116.

Cayco-Gajic, N Alex and R Angus Silver (2019). "Re-evaluating circuit mechanisms underlying pattern separation". In: *Neuron* 101.4, pp. 584–602.

Cerminara, Nadia L, Richard Apps, and Dilwyn E Marple-Horvat (2009). "An internal model of a moving visual target in the lateral cerebellum". In: *The Journal of physiology* 587.2, pp. 429–442.

Chen, Beth L, David H Hall, and Dmitri B Chklovskii (2006). "Wiring optimization can relate neuronal structure and function". In: *Proceedings of the National Academy of Sciences* 103.12, pp. 4723–4728.

Chen, Jingwen et al. (2024). "Flexible control of sequence working memory in the macaque frontal cortex". In: *Neuron* 112.20, pp. 3502–3514.

Cherniak, Christopher et al. (2004). "Global optimization of cerebral cortex layout". In: *Proceedings of the National Academy of Sciences* 101.4, pp. 1081–1086.

Chintaluri, Chaitanya and Tim P Vogels (2023). "Metabolically regulated spiking could serve neuronal energy homeostasis and protect from reactive oxygen species". In: *Proceedings of the National Academy of Sciences* 120.48, e2306525120.

Chklovskii, Dmitri B, Thomas Schikorski, and Charles F Stevens (2002). "Wiring optimization in cortical circuits". In: *Neuron* 34.3, pp. 341–347.

Cho, Youngmin and Lawrence Saul (2009). "Kernel methods for deep learning". In: *Advances in neural information processing systems* 22.

Cohen, Jeremy E and Nicolas Gillis (2019). "Identifiability of complete dictionary learning". In: *SIAM Journal on Mathematics of Data Science* 1.3, pp. 518–536.

Comon, Pierre (1994). "Independent component analysis, a new concept?" In: *Signal processing* 36.3, pp. 287–314.

Compte, Albert et al. (2000). "Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model". In: *Cerebral cortex* 10.9, pp. 910–923.

Conklin, John and Chris Eliasmith (2005). "A controlled attractor network model of path integration in the rat". In: *Journal of computational neuroscience* 18, pp. 183–203.

Constantinescu, Alexandra O, Jill X O'Reilly, and Timothy EJ Behrens (2016). "Organizing conceptual knowledge in humans with a gridlike code". In: *Science* 352.6292, pp. 1464–1468.

Crowe, David A et al. (2014). "Dynamic representation of the temporal and sequential structure of rhythmic movements in the primate medial premotor cortex". In: *Journal of Neuroscience* 34.36, pp. 11972–11983.

Cueva, Christopher J and Xue-Xin Wei (2018). "Emergence of grid-like representations by training recurrent neural networks to perform spatial localization". In: *arXiv preprint arXiv:1803.07770*.

Curtis, Clayton E and Mark D'esposito (2004). "The effects of prefrontal lesions on working memory performance and theory". In: *Cognitive, Affective, & Behavioral Neuroscience* 4.4, pp. 528–539.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

D'Angelo, Egidio (2014). "The organization of plasticity in the cerebellar cortex: from synapses to control". In: *Progress in brain research* 210, pp. 31–58.

D'Esposito, Mark and Bradley R Postle (2015). "The cognitive neuroscience of working memory". In: *Annual review of psychology* 66.1, pp. 115–142.

Deighton, Jared et al. (2024). "Higher-Order Spatial Information for Self-Supervised Place Cell Learning". In: *arXiv preprint arXiv:2407.06195*.

Dey, Nolan et al. (2025). "Neuron-based explanations of neural networks sacrifice completeness and interpretability". In: *Transactions on Machine Learning Research*. URL: https://openreview.net/forum?id=UWNa9Pv6qA.

Dietrich, Robin et al. (2024). "Grid codes vs. multi-scale, multi-field place codes for space". In: *Frontiers in Computational Neuroscience* 18, p. 1276292.

Djolonga, Josip (2020). *torch-two-sample*. https://github.com/josipd/torch-two-sample.

Donoho, David and Victoria Stodden (2003). "When does non-negative matrix factorization give a correct decomposition into parts?" In: *Advances in neural information processing systems* 16.

Dordek, Yedidyah et al. (2016). "Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis". In: *Elife* 5, e10094.

Dorrell, Will et al. (2023). "Actionable Neural Representations: Grid Cells from Minimal Constraints". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=xfqDe72zh41.

Dorrell, William, Kyle Hsu, et al. (2025). "Range, not Independence, Drives Modularity in Biologically Inspired Representations". In: *The Thirteenth International Conference on Learning Representations*.

Dorrell, William, Maria Yuffa, and Peter Latham (2023). "Meta-Learning the Inductive Biases of Simple Neural Circuits". In: *ICML*.

Driscoll, Laura, Krishna Shenoy, and David Sussillo (2022). "Flexible multitask computation in recurrent networks utilizes shared dynamical motifs". In: *bioRxiv*, pp. 2022–08.

Dunn, Benjamin et al. (2017). "Grid cells show field-to-field variability and this explains the aperiodic response of inhibitory interneurons". In: *arXiv preprint arXiv:1701.04893*.

Dür, Mirjam (2010). "Copositive programming–a survey". In: *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization*. Springer, pp. 3–20.

Elhage, Nelson et al. (2022). "Toy Models of Superposition". In: *arXiv preprint arXiv:2209.10652*.

Eliav, Tamir et al. (2021). "Multiscale representation of very large environments in the hippocampus of flying bats". In: *Science* 372.6545, eabg4020.

Ellis, Kaitlyn Elizabeth et al. (2023). "Evolution of connectivity architecture in the Drosophila mushroom body". In: *bioRxiv*, pp. 2023–02.

Ellis, Kevin et al. (2020). "Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning". In: *arXiv preprint arXiv:2006.08381*.

Elsayed, Gamaleldin F et al. (2016). "Reorganization between preparatory and movement population responses in motor cortex". In: *Nature communications* 7.1, p. 13239.

Ergen, Tolga and Mert Pilanci (2021). "Revealing the structure of deep neural networks via convex duality". In: *International Conference on Machine Learning*. PMLR, pp. 3004–3014.

Euler, Thomas et al. (2014). "Retinal bipolar cells: elementary building blocks of vision". In: *Nature Reviews Neuroscience* 15.8, pp. 507–519.

Fenton, André A et al. (2008). "Unmasking the CA1 ensemble place code by exposures to small and large environments: more place cells and multiple, irregularly arranged, and expanded place fields in the larger space". In: *Journal of Neuroscience* 28.44, pp. 11250–11262.

Finkelstein, Arseny et al. (2015). "Three-dimensional head-direction coding in the bat brain". In: *Nature* 517.7533, pp. 159–164.

Fu, Xiao, Kejun Huang, and Nicholas D Sidiropoulos (2018). "On identifiability of nonnegative matrix factorization". In: *IEEE Signal Processing Letters* 25.3, pp. 328–332.

Fulton, William and Joe Harris (1991). *Representation Theory: A First Course*. Springer-Verlag.

Funahashi, Shintaro, Matthew V Chafee, and Patricia S Goldman-Rakic (1993). "Prefrontal neuronal activity in rhesus monkeys performing a delayed anti-saccade task". In: *Nature* 365.6448, pp. 753–756.

Fuster, Joaquin M and Garrett E Alexander (1971). "Neuron activity related to short-term memory". In: *Science* 173.3997, pp. 652–654.

El-Gaby, Mohamady et al. (2024). "A cellular basis for mapping behavioural structure". In: *Nature*, pp. 1–10.

Gao, Ruiqi, Jianwen Xie, Xue-Xin Wei, et al. (2021). "On Path Integration of grid cells: isotropic metric, conformal embedding and group representation". In: *Advances in neural information processing systems* 34.

Gao, Ruiqi, Jianwen Xie, Song-Chun Zhu, et al. (2019). "Learning grid cells as vector representation of self-position coupled with matrix representation of self-motion". In: *Internal Conference on Learning Representations*.

Gardner, Richard J et al. (2022). "Toroidal topology of population activity in grid cells". In: *Nature* 602.7895, pp. 123–128.

Gauthier, Jon and Roger Levy (2019). "Linking artificial and human neural representations of language". In: *arXiv preprint arXiv:1910.01244*.

Gautrais, Jacques and Simon Thorpe (1998). "Rate coding versus temporal order coding: a theoretical approach". In: *Biosystems* 48.1-3, pp. 57–65.

Gil, Mariana et al. (2018). "Impaired path integration in mice with disrupted grid cell firing". In: *Nature neuroscience* 21.1, pp. 81–91.

Gillis, Nicolas and Robert Luce (2024). "Checking the sufficiently scattered condition using a global non-convex optimization software". In: *IEEE Signal Processing Letters* 31, pp. 1610–1614.

Ginosar, Gily et al. (2021). "Locally ordered representation of 3D space in the entorhinal cortex". In: *Nature* 596.7872, pp. 404–409.

Giocomo, Lisa M, May-Britt Moser, and Edvard I Moser (2011). "Computational models of grid cells". In: *Neuron* 71.4, pp. 589–603.

Gjorgjieva, Julijana, Haim Sompolinsky, and Markus Meister (2014). "Benefits of pathway splitting in sensory coding". In: *Journal of Neuroscience* 34.36, pp. 12127–12144.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 249–256.

Goldstein, Ariel et al. (2022). "Shared computational principles for language processing in humans and deep language models". In: *Nature neuroscience* 25.3, pp. 369–380.

Golub, Gene H and Victor Pereyra (1973). "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate". In: *SIAM Journal on numerical analysis* 10.2, pp. 413–432.

Goodman, Dan et al. (2022). *Spiking Neural Network Models in Neuroscience - Cosyne Tutorial 2022*. https://zenodo.org/record

Gretton, Arthur et al. (2012). "A kernel two-sample test". In: *The Journal of Machine Learning Research* 13.1, pp. 723–773.

Grieves, Roddy M et al. (2021). "Irregular distribution of grid cell firing fields in rats exploring a 3D volumetric space". In: *Nature neuroscience* 24.11, pp. 1567–1573.

Gunasekar, Suriya et al. (2017). "Implicit regularization in matrix factorization". In: *Advances in Neural Information Processing Systems* 30.

Gupta, Divyansh et al. (2023). "Panoramic visual statistics shape retina-wide organization of receptive fields". In: *Nature Neuroscience* 26.4, pp. 606–614.

Gutiérrez-Guzmán, Blanca E, J Jesús Hernández-Pérez, and Holger Dannenberg (2025). "Tiling of large-scaled environments by grid cells requires experience". In: *bioRxiv*.

Hafting, Torkel et al. (2005). "Microstructure of a spatial map in the entorhinal cortex". In: *Nature* 436.7052, pp. 801–806.

Hägglund, Martin et al. (2019). "Grid-cell distortion along geometric borders". In: *Current Biology* 29.6, pp. 1047–1054.

Hansen, Jakob and Robert Ghrist (2019). "Toward a spectral theory of cellular sheaves". In: *Journal of Applied and Computational Topology* 3.4, pp. 315–358.

Hardcastle, Kiah et al. (2017). "A multiplexed, heterogeneous, and adaptive code for navigation in medial entorhinal cortex". In: *Neuron* 94.2, pp. 375–387.

Hardt, Moritz, Ben Recht, and Yoram Singer (2016). "Train faster, generalize better: Stability of stochastic gradient descent". In: *International conference on machine learning*. PMLR, pp. 1225–1234.

Harland, Bruce et al. (2021). "Dorsal CA1 hippocampal place cells form a multi-scale representation of megaspace". In: *Current Biology* 31.10, pp. 2178–2190.

Harris, Julia J, Renaud Jolivet, and David Attwell (2012). "Synaptic energy use and supply". In: *Neuron* 75.5, pp. 762–777.

Harris, Kameron Decker (2019). "Additive function approximation in the brain". In: *arXiv preprint arXiv:1909.02603*.

Hayashi, Tatsuya Tatz et al. (2022). "Mushroom body input connections form independently of sensory activity in Drosophila melanogaster". In: *Current Biology* 32.18, pp. 4000–4012.

Hayman, Robin MA et al. (2015). "Grid cells on steeply sloping terrain: evidence for planar rather than volumetric encoding". In: *Frontiers in psychology* 6, p. 925.

Hige, Toshihide (2018). "What can tiny mushrooms in fruit flies tell us about learning and memory?" In: *Neuroscience Research* 129, pp. 8–16.

Hillar, Christopher J and Friedrich T Sommer (2015). "When can dictionary learning uniquely recover sparse data from subsamples?" In: *IEEE Transactions on Information Theory* 61.11, pp. 6290–6297.

Hinton, Geoffrey et al. (2006). "Unsupervised discovery of nonlinear structure using contrastive backpropagation". In: *Cognitive science* 30.4, pp. 725–731.

Hiratani, Naoki and Peter E Latham (2022). "Developmental and evolutionary constraints on olfactory circuit selection". In: *Proceedings of the National Academy of Sciences* 119.11, e2100600119.

Horan, Daniella, Eitan Richardson, and Yair Weiss (2021). "When is unsupervised disentanglement possible?" In: *Advances in Neural Information Processing Systems* 34, pp. 5150–5161.

Høydal, Øyvind Arne et al. (Apr. 2019). "Object-vector coding in the medial entorhinal cortex". In: *Nature* 568.7752, pp. 400–404. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1077-7. URL: http://dx.doi.org/10.1038/s41586-019-1077-7.

Hsu, Kyle, William Dorrell, et al. (2023). "Disentanglement via latent quantization". In: *Advances in Neural Information Processing Systems* 36.

Hsu, Kyle, Jubayer Ibn Hamid, et al. (2024). "Tripod: Three Complementary Inductive Biases for Disentangled Representation Learning". In: *International Conference on Machine Learning*.

Hu, Hong and Yue M Lu (2022). "Universality laws for high-dimensional learning with random features". In: *IEEE Transactions on Information Theory* 69.3, pp. 1932–1964.

Hu, Jingzhou and Kejun Huang (2023). "Global Identifiability of L1-based Dictionary Learning via Matrix Volume Optimization". In: *Advances in Neural Information Processing Systems* 36, pp. 36165–36186.

Huang, Kejun, Nicholas D Sidiropoulos, and Ananthram Swami (2013). "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition". In: *IEEE Transactions on Signal Processing* 62.1, pp. 211–224.

Hubel, David H and Torsten N Wiesel (1962). "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1, p. 106.

Huber, David E (2025). "A memory model of rodent spatial navigation in which place cells are memories arranged in a grid and grid cells are non-spatial". In: *eLife* 13, RP95733.

Hulse, Brad K and Vivek Jayaraman (2020). "Mechanisms underlying the neural computation of head direction". In: *Annual review of neuroscience* 43.1, pp. 31–54.

Hume, David (1748). *An enquiry concerning human understanding and other writings*.

Hyvarinen, Aapo, Ilyes Khemakhem, and Hiroshi Morioka (2023). "Nonlinear Independent Component Analysis for Principled Disentanglement in Unsupervised Deep Learning". In: *arXiv preprint arXiv:2303.16535*.

Issa, John B and Kechen Zhang (2012). "Universal conditions for exact path integration in neural systems". In: *Proceedings of the National Academy of Sciences* 109.17, pp. 6716–6720.

Ito, Masao (1972). "Neural design of the cerebellar motor control system". In: *Brain research* 40.1, pp. 81–84.

Ivanic, Joseph and Klaus Ruedenberg (1996). "Rotation matrices for real spherical harmonics. Direct determination by recursion". In: *The Journal of Physical Chemistry* 100.15, pp. 6342–6347.

— (1998). "Rotation matrices for real spherical harmonics. direct determination by recursion". In: *The Journal of Physical Chemistry A* 102.45, pp. 9099–9100.

Jacob, Pierre-Yves et al. (2019). "Path integration maintains spatial periodicity of grid cell firing in a 1D circular track". In: *Nature communications* 10.1, p. 840.

Jacot, Arthur, Franck Gabriel, and Clément Hongler (2018). "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems* 31.

Jarvis, Devon, Richard Klein, Benjamin Rosman, and Andrew Saxe (2023). "On the specialization of neural modules". In: *The Eleventh International Conference on Learning Representations*.

Jarvis, Devon, Richard Klein, Benjamin Rosman, and Andrew M Saxe (2025). "Make haste slowly: A theory of emergent structured mixed selectivity in feature learning reLU networks". In: *arXiv preprint arXiv:2503.06181*.

Jones, Judson P and Larry A Palmer (1987). "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex". In: *Journal of neurophysiology* 58.6, pp. 1233–1258.

Jun, Heechul et al. (2020). "Disrupted place cell remapping and impaired grid cells in a knockin model of Alzheimer's disease". In: *Neuron* 107.6, pp. 1095–1112.

Kang, Louis and Vijay Balasubramanian (2019). "A geometric attractor mechanism for self-organization of entorhinal grid modules". In: *Elife* 8, e46687.

Kang, Yul HR, Daniel M Wolpert, and Máté Lengyel (2023). "Spatial uncertainty and environmental geometry in navigation". In: *bioRxiv*, pp. 2023–01.

Kanwisher, Nancy and Galit Yovel (2006). "The fusiform face area: a cortical region specialized for the perception of faces". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 361.1476, pp. 2109–2128.

Kaufman, Matthew T et al. (2016). "The largest response component in the motor cortex reflects movement timing but not movement type". In: *eneuro* 3.4.

Kawato, Mitsuo et al. (2021). "50 years since the Marr, Ito, and Albus models of the cerebellum". In: *Neuroscience* 462, pp. 151–174.

Kell, Alexander JE et al. (2018). "A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy". In: *Neuron* 98.3, pp. 630–644.

Kennedy, Ann et al. (2014). "A temporal basis for predicting the sensory consequences of motor commands in an electric fish". In: *Nature neuroscience* 17.3, pp. 416–422.

Khemakhem, Ilyes et al. (2020). "Variational autoencoders and nonlinear ica: A unifying framework". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 2207–2217.

Khona, Mikail, Sarthak Chandra, and Ila Fiete (2025). "Global modules robustly emerge from local interactions and smooth gradients". In: *Nature*, pp. 1–10.

Kim, Sung Soo et al. (2017). "Ring attractor dynamics in the Drosophila central brain". In: *Science* 356.6340, pp. 849–853.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

Kivva, Bohdan et al. (2022). "Identifiability of deep generative models without auxiliary information". In: *Advances in Neural Information Processing Systems* 35, pp. 15687–15701.

Klindt, David A. et al. (2021). "Towards Nonlinear Disentanglement in Natural Data with Temporal Sparse Coding". In: *International Conference on Learning Representations*.

Klukas, Mirko, Marcus Lewis, and Ila Fiete (2020). "Efficient and flexible representation of higher-dimensional cognitive variables with grid cells". In: *PLoS computational biology* 16.4, e1007796.

Knapp, Anthony W. (2002). *Lie Groups Beyond an Introduction*. Springer.

Knogler, Laura D et al. (2017). "Sensorimotor representations in cerebellar granule cells in larval zebrafish are dense, spatially organized, and non-temporally patterned". In: *Current Biology* 27.9, pp. 1288–1302.

Koulakov, Alexei A and Dmitri B Chklovskii (2001). "Orientation preference patterns in mammalian visual cortex: a wire length minimization approach". In: *Neuron* 29.2, pp. 519–527.

Kriegeskorte, Nikolaus, Marieke Mur, and Peter A Bandettini (2008). "Representational similarity analysis-connecting the branches of systems neuroscience". In: *Frontiers in systems neuroscience* 2, p. 249.

Krupic, Julija et al. (2015). "Grid cell symmetry is shaped by environmental geometry". In: *Nature* 518.7538, pp. 232–235.

Kubie, John L and André A Fenton (2012). "Linear look-ahead in conjunctive cells: an entorhinal mechanism for vector-based navigation". In: *Frontiers in neural circuits* 6, p. 20.

Kubota, Kisou and Hiroaki Niki (1971). "Prefrontal cortical unit activity and delayed alternation performance in monkeys." In: *Journal of neurophysiology* 34.3, pp. 337–347.

Kunz, Lukas et al. (2015). "Reduced grid-cell–like representations in adults at genetic risk for Alzheimer's disease". In: *Science* 350.6259, pp. 430–433.

Lachapelle, Sébastien and Simon Lacoste-Julien (2022). "Partial disentanglement via mechanism sparsity". In: *arXiv preprint arXiv:2207.07732*.

Lachapelle, Sébastien, Pau Rodriguez, et al. (2022). "Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA". In: *Conference on Causal Learning and Reasoning*. PMLR, pp. 428–484.

Lange, Robert Tjarko (2022). "evosax: JAX-based Evolution Strategies". In: *arXiv preprint arXiv:2212.04180*.

Lange, Robert Tjarko et al. (2022). "Discovering Evolution Strategies via Meta-Black-Box Optimization". In: *arXiv preprint arXiv:2211.11260*.

Laughlin, Simon (1981). "A simple coding procedure enhances a neuron's information capacity". In: *Zeitschrift für Naturforschung c* 36.9-10, pp. 910–912.

LeCun, Yann (1998). "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/*.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lee, Jae Sung et al. (2020). "The statistical structure of the hippocampal code for space as a function of time, context, and value". In: *Cell* 183.3, pp. 620–635.

Li, Michael Y, Erin Grant, and Thomas L Griffiths (2021). "Meta-learning inductive biases of learning systems with Gaussian processes". In: *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*.

Lillicrap, Timothy P et al. (2020). "Backpropagation and the brain". In: *Nature Reviews Neuroscience* 21.6, pp. 335–346.

Lin, Andrew C et al. (2014). "Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination". In: *Nature neuroscience* 17.4, pp. 559–568.

Linderman, Scott, Matthew Johnson, et al. (2017). "Bayesian learning and inference in recurrent switching linear dynamical systems". In: *Artificial intelligence and statistics*. PMLR, pp. 914–922.

Linderman, Scott, Annika Nichols, et al. (2019). "Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans". In: *BioRxiv*, p. 621540.

Lindsay, Grace W (2021). "Convolutional neural networks as a model of the visual system: Past, present, and future". In: *Journal of cognitive neuroscience* 33.10, pp. 2017–2031.

Litwin-Kumar, Ashok, Kameron Decker Harris, et al. (2017). "Optimal degrees of synaptic connectivity". In: *Neuron* 93.5, pp. 1153–1164.

Litwin-Kumar, Ashok and Srinivas C Turaga (2019). "Constraining computational models using electron microscopy wiring diagrams". In: *Current opinion in neurobiology* 58, pp. 94–100.

Logiaco, Laureline, LF Abbott, and Sean Escola (2021). "Thalamic control of cortical dynamics in a model of flexible motor sequencing". In: *Cell reports* 35.9.

Logothetis, Nikos K, Jon Pauls, and Tomaso Poggio (1995). "Shape representation in the inferior temporal cortex of monkeys". In: *Current biology* 5.5, pp. 552–563.

Ma, Tzuhsuan (2020). "Towards a theory for the emergence of grid and place cell codes". PhD thesis. Massachusetts Institute of Technology.

Mairal, Julien and Jean-Philippe Vert (2018). "Machine learning with kernel methods". In: *Lecture Notes, January* 10.

Mancoo, Allan, Sander Keemink, and Christian K Machens (2020). "Understanding spiking networks through convex optimization". In: *Advances in neural information processing systems* 33, pp. 8824–8835.

Mandt, Stephan, Matthew D Hoffman, and David M Blei (2017). "Stochastic gradient descent as approximate bayesian inference". In: *arXiv preprint arXiv:1704.04289*.

Marr, David (1969). "A theory of cerebral cortex". In: *Journal of Physiology* 202.2, pp. 437–470.

Martín-Sánchez, Guillermo, Christian K Machens, and William F Podlaski (2025). "Three types of remapping with linear decoders: a population-geometric perspective". In: *bioRxiv*, pp. 2025–03.

Masset, Paul et al. (2022). "Natural gradient enables fast sampling in spiking neural networks". In: *Advances in neural information processing systems* 35, pp. 22018–22034.

Mathis, Alexander, Andreas VM Herz, and Martin Stemmler (2012). "Optimal population codes for space: grid cells outperform place cells". In: *Neural computation* 24.9, pp. 2280–2317.

Mathis, Alexander, Andreas VM Herz, and Martin B Stemmler (2012). "Resolution of nested neuronal representations can be exponential in the number of neurons". In: *Physical review letters* 109.1, p. 018103.

McClelland, James L, Bruce L McNaughton, and Randall C O'Reilly (1995). "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." In: *Psychological review* 102.3, p. 419.

McNaughton, Bruce L et al. (2006). "Path integration and the neural basis of the'cognitive map'". In: *Nature Reviews Neuroscience* 7.8, pp. 663–678.

Merchant, Hugo and Bruno B Averbeck (2017). "The computational and neural basis of rhythmic timing in medial premotor cortex". In: *Journal of Neuroscience* 37.17, pp. 4552–4564.

Merchant, Hugo, Deborah L Harrington, and Warren H Meck (2013). "Neural basis of the perception and estimation of time". In: *Annual review of neuroscience* 36, pp. 313–336.

Merchant, Hugo and Victor de Lafuente (2024). "A second introduction to the neurobiology of interval timing". In: *Neurobiology of Interval Timing*, pp. 3–23.

Merchant, Hugo, Germán Mendoza, et al. (2023). "Different time encoding strategies within the medial premotor areas of the primate". In: *bioRxiv*, pp. 2023–01.

Merchant, Hugo, Oswaldo Pérez, et al. (2013). "Interval tuning in the primate medial premotor cortex as a general timing mechanism". In: *Journal of Neuroscience* 33.21, pp. 9082–9096.

Miall, RC (2007). "The cerebellum, predictive control and motor coordination". In: *Novartis Foundation Symposium 218-Sensory Guidance of Movement: Sensory Guidance of Movement: Novartis Foundation Symposium 218*. Wiley Online Library, pp. 272–290.

Micchelli, Charles A, Yuesheng Xu, and Haizhang Zhang (2006). "Universal Kernels." In: *Journal of Machine Learning Research* 7.12.

Modi, Mehrab N, Yichun Shuai, and Glenn C Turner (2020). "The Drosophila mushroom body: from architecture to algorithm in a learning circuit". In: *Annual review of neuroscience* 43.1, pp. 465–484.

Mok, Robert M and Bradley C Love (2019). "A non-spatial account of place and grid cells based on clustering models of concept learning". In: *Nature communications* 10.1, p. 5685.

Montavon, Grégoire, Wojciech Samek, and Klaus-Robert Müller (2018). "Methods for interpreting and understanding deep neural networks". In: *Digital signal processing* 73, pp. 1–15.

Moran, Gemma Elyse et al. (2022). "Identifiable Deep Generative Models via Sparse Decoding". In: *Transactions on Machine Learning Research*.

Mosheiff, Noga et al. (2017). "An efficient coding theory for a dynamic trajectory predicts non-uniform allocation of entorhinal grid cells to modules". In: *PLoS computational biology* 13.6, e1005597.

Müller, Claus (2012). *Analysis of spherical symmetries in Euclidean spaces*. Vol. 129. Springer Science & Business Media.

Mushiake, Hajime et al. (2006). "Activity in the lateral prefrontal cortex reflects multiple steps of future events in action plans". In: *Neuron* 50.4, pp. 631–641.

Nakajima, Toshi et al. (2013). "Two-dimensional representation of action and arm-use sequences in the presupplementary and supplementary motor areas". In: *Journal of neuroscience* 33.39, pp. 15533–15544.

Nassar, Josue et al. (2018). "Tree-structured recurrent switching linear dynamical systems for multi-scale modeling". In: *arXiv preprint arXiv:1811.12386*.

Neftci, Emre O, Hesham Mostafa, and Friedemann Zenke (2019). "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks". In: *IEEE Signal Processing Magazine* 36.6, pp. 51–63.

Nguyen, Tri M et al. (2023). "Structured cerebellar connectivity supports resilient pattern separation". In: *Nature* 613.7944, pp. 543–549.

Nieuwenbuys, R and Charles Nicholson (1969). *Aspects of the histology of the cerebellum of mormyrid fishes*.

Nieuwenhuys, Rudolf, Hans J Ten Donkelaar, and Charles Nicholson (2014). *The central nervous system of vertebrates*. Springer.

O'Keefe, John and Jonathan Dostrovsky (1971). "The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat." In: *Brain research*.

Ocko, Samuel A et al. (2018). "Emergent elasticity in the neural code for space". In: *Proceedings of the National Academy of Sciences* 115.50, E11798–E11806.

Ohbayashi, Machiko, Kenichi Ohki, and Yasushi Miyashita (2003). "Conversion of working memory to motor sequence in the monkey premotor cortex". In: *Science* 301.5630, pp. 233–236.

Okajima, Kenji (1998a). "The Gabor function extracts the maximum information from input local signals". In: *Neural Networks* 11.3, pp. 435–439.

— (1998b). "Two-dimensional Gabor-type receptive field as derived by mutual information maximization". In: *Neural Networks* 11.3, pp. 441–447.

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature visualization". In: *Distill* 2.11, e7.

Olshausen, Bruno A and David J Field (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583, pp. 607–609.

Paccanaro, Alberto and Geoffrey E Hinton (2000). "Learning Distributed Representations by Mapping Concepts and Relations into a Linear Space." In: *ICML*, pp. 711–718.

— (2001). "Learning hierarchical structures with linear relational embedding". In: *Advances in neural information processing systems* 14.

— (2002). "Learning distributed representations of relational data using linear relational embedding". In: *Neural Nets WIRN Vietri-01: Proceedings of the 12th Italian Workshop on Neural Nets, Vietri sul Mare, Salerno, Italy, 17–19 May 2001*. Springer, pp. 134–143.

Pandey, Biraj et al. (2021). "Structured random receptive fields enable informative sensory encodings". In: *bioRxiv*.

Panichello, Matthew F and Timothy J Buschman (2021). "Shared mechanisms underlie the control of working memory and attention". In: *Nature* 592.7855, pp. 601–605.

Pehlevan, Cengiz, Sreyas Mohan, and Dmitri B Chklovskii (2017). "Blind nonnegative source separation using biological neural networks". In: *Neural computation* 29.11, pp. 2925–2954.

Penfield, Wilder and JP Evans (1935). "The frontal lobe in man: A clinical study of maximum removals." In: *Brain: A Journal of Neurology*.

Perez, Ethan et al. (2018). "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Perge, János A et al. (2012). "Why do axons differ in caliber?" In: *Journal of Neuroscience* 32.2, pp. 626–638.

Petrucco, Luigi et al. (2023). "Neural dynamics and architecture of the heading direction circuit in zebrafish". In: *Nature neuroscience* 26.5, pp. 765–773.

Pettersen, Markus et al. (2024). "Self-supervised grid cells without path integration". In: *bioRxiv*, pp. 2024–05.

Pilanci, Mert and Tolga Ergen (2020). "Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks". In: *International Conference on Machine Learning*. PMLR, pp. 7695–7705.

Piwek, Emilia P, Mark G Stokes, and Christopher Summerfield (2023). "A recurrent neural network model of prefrontal brain activity during a working memory task". In: *PLoS Computational Biology* 19.10, e1011555.

Podlaski, William F and Christian K Machens (2024). "Approximating nonlinear functions with latent boundaries in low-rank excitatory-inhibitory spiking networks". In: *Neural Computation* 36.5, pp. 803–857.

Politis, Archontis et al. (2016). "Microphone array processing for parametric spatial audio techniques". In.

Pouget, Alexandre and Terrence J Sejnowski (1997). "Spatial transformations in the parietal cortex using basis functions". In: *Journal of cognitive neuroscience* 9.2, pp. 222–237.

Ravsky, Alex (2025). *Bound on the min of the sum of cosines at two rationally related frequencies*. URL: `https://math.stackexchange.com/questions/4966081/bound-on-the-min-of-the-sum-of-cosines-at-two-rationally-related-frequencies`. (accessed: 24.05.2025).

Rebecca, RG et al. (2025). "Spatial periodicity in grid cell firing is explained by a neural sequence code of 2-D trajectories". In: *eLife* 13, RP96627.

Redish, A David, Adam N Elga, and David S Touretzky (1996). "A coupled attractor model of the rodent head direction system". In: *Network: computation in neural systems* 7.4, p. 671.

Redman, William T et al. (2025). "Robust variability of grid cell properties within individual grid modules enhances encoding of local space". In: *Elife* 13, RP100652.

Rezende, Danilo Jimenez and Fabio Viola (2018). "Taming vaes". In: *arXiv preprint arXiv:1810.00597*.

Rich, P Dylan, Hua-Peng Liaw, and Albert K Lee (2014). "Large environments reveal the statistical structure governing hippocampal representations". In: *Science* 345.6198, pp. 814–817.

Richards, Blake A et al. (2019). "A deep learning framework for neuroscience". In: *Nature neuroscience* 22.11, pp. 1761–1770.

Rigotti, Mattia et al. (2013). "The importance of mixed selectivity in complex cognitive tasks". In: *Nature* 497.7451, pp. 585–590.

Roth, Karsten et al. (2023). "Disentanglement of Correlated Factors via Hausdorff Factorized Support". In: *The Eleventh International Conference on Learning Representations*.

Ruhe, Axel (1970). "Perturbation bounds for means of eigenvalues and invariant subspaces". In: *BIT Numerical Mathematics* 10.3, pp. 343–354.

Rumelhart, David E and James L McClelland (1985). *On learning the past tenses of English verbs*. Tech. rep.

Sacks, Oliver (2011). *The man who mistook his wife for a hat: And other clinical tales*. Brilliance Audio.

Sahiner, Arda et al. (2020). "Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms". In: *arXiv preprint arXiv:2012.13329*.

Salinas, Emilio and LF Abbott (1994). "Vector reconstruction from firing rates". In: *Journal of computational neuroscience* 1.1, pp. 89–107.

Samborska, Veronika et al. (2022). "Complementary task representations in hippocampus and prefrontal cortex for generalizing the structure of problems". In: *Nature Neuroscience* 25.10, pp. 1314–1326.

Samsonovich, Alexei and Bruce L McNaughton (1997). "Path integration and cognitive mapping in a continuous attractor neural network model". In: *Journal of Neuroscience* 17.15, pp. 5900–5920.

Sargolini, Francesca et al. (2006). "Conjunctive representation of position, direction, and velocity in entorhinal cortex". In: *Science* 312.5774, pp. 758–762.

Saxe, Andrew, Shagun Sodhani, and Sam Jay Lewallen (2022). "The neural race reduction: Dynamics of abstraction in gated networks". In: *International Conference on Machine Learning*. PMLR, pp. 19287–19309.

Saxe, Andrew M, James L McClelland, and Surya Ganguli (2019). "A mathematical theory of semantic development in deep neural networks". In: *Proceedings of the National Academy of Sciences* 116.23, pp. 11537–11546.

Schaeffer, Rylan, Mikail Khona, Adrian Bertagnoli, et al. (2023). "Testing assumptions underlying a unified theory for the origin of grid cells". In: *arXiv preprint arXiv:2311.16295*.

Schaeffer, Rylan, Mikail Khona, and Ila Fiete (2022). "No Free Lunch from Deep Learning in Neuroscience: A Case Study through Models of the Entorhinal-Hippocampal Circuit". In: *ICML 2022 Workshop AI4Science*.

Schaeffer, Rylan, Mikail Khona, Sanmi Koyejo, et al. (2023). "Disentangling Fact from Grid Cell Fiction in Trained Deep Path Integrators". In: *ArXiv*, arXiv–2312.

Schaeffer, Rylan, Mikail Khona, Tzuhsuan Ma, et al. (2023). "Self-supervised learning of representations for space generates multi-modular grid cells". In: *Advances in Neural Information Processing Systems* 36, pp. 23140–23157.

Schaffer, Evan S et al. (2018). "Odor perception on the two sides of the brain: consistency despite randomness". In: *Neuron* 98.4, pp. 736–742.

Schimel, Marine, Ta-Chu Kao, and Guillaume Hennequin (2024). "When and why does motor preparation arise in recurrent neural network models of motor control?" In: *Elife* 12, RP89131.

Schmutz, Valentin, Johanni Brea, and Wulfram Gerstner (2025). "Emergent rate-based dynamics in duplicate-free populations of spiking neurons". In: *Physical Review Letters* 134.1, p. 018401.

Schoenberg, Isaac J (1942). "Positive definite functions on spheres". In.

Schøyen, Vemund et al. (2023). "Coherently remapping toroidal cells but not grid cells are responsible for path integration in virtual agents". In: *Iscience* 26.11.

Schøyen, Vemund Sigmundson et al. (2025). "Hexagons all the way down: Grid cells as a conformal isometric map of space". In: *PLOS Computational Biology* 21.2, e1012804.

Schuck, Nicolas W et al. (2016). "Human orbitofrontal cortex represents a cognitive map of state space". In: *Neuron* 91.6, pp. 1402–1412.

Schug, Simon et al. (2024). "Discovering modular solutions that generalize compositionally". In: *The Twelfth International Conference on Learning Representations*.

Schwartenbeck, Philipp et al. (2023). "Generative replay underlies compositional inference in the hippocampal-prefrontal circuit". In: *Cell* 186.22, pp. 4885–4897.

Seelig, Johannes D and Vivek Jayaraman (2015). "Neural dynamics for landmark orientation and angular path integration". In: *Nature* 521.7551, pp. 186–191.

Sengupta, Anirvan et al. (2018). "Manifold-tiling localized receptive fields are optimal in similarity-preserving neural networks". In: *Advances in neural information processing systems* 31.

Seri, Bettina et al. (2001). "Astrocytes give rise to new neurons in the adult mammalian hippocampus". In: *Journal of Neuroscience* 21.18, pp. 7153–7160.

Shadmehr, Reza (2020). "Population coding in the cerebellum: a machine learning perspective". In: *Journal of neurophysiology* 124.6, pp. 2022–2051.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *The Bell system technical journal* 27.3, pp. 379–423.

Shima, Keisetsu, Masaki Isoda, et al. (2007). "Categorization of behavioural sequences in the prefrontal cortex". In: *Nature* 445.7125, pp. 315–318.

Shima, Keisetsu and Jun Tanji (2000). "Neuronal activity in the supplementary and presupplementary motor areas for temporal organization of multiple movements". In: *Journal of neurophysiology* 84.4, pp. 2148–2160.

Simmons, Robert E and Res Altwegg (2010). "Necks-for-sex or competing browsers? A critique of ideas on the evolution of giraffe". In: *Journal of zoology* 282.1, pp. 6–12.

Simon, James B, Madeline Dickens, and Michael R DeWeese (2021). "Neural tangent kernel eigenvalues accurately predict generalization". In: *arXiv preprint arXiv:2110.03922*.

Skaggs, William et al. (1994). "A model of the neural basis of the rat's sense of direction". In: *Advances in neural information processing systems* 7.

Sollich, Peter (1998). "Learning curves for Gaussian processes". In: *Advances in neural information processing systems* 11.

Soni, Aneri and Michael J Frank (2025). "Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit". In: *eLife* 13, RP97894.

Sorscher, Ben, Surya Ganguli, and Haim Sompolinsky (2022). "Neural representational geometry underlies few-shot concept learning". In: *Proceedings of the National Academy of Sciences* 119.43, e2200800119.

Sorscher, Ben, Gabriel Mel, et al. (2019). "A unified theory for the origin of grid cells through the lens of pattern formation". In: *Advances in neural information processing systems* 32.

Sorscher, Ben, Gabriel C Mel, Aran Nayebi, et al. (2022). "When and why grid cells appear or not in trained path integrators". In: *bioRxiv*.

Sorscher, Ben, Gabriel C Mel, Samuel A Ocko, et al. (2023). "A unified theory for the computational and mechanistic origins of grid cells". In: *Neuron* 111.1, pp. 121–137.

Sprekeler, Henning, Tiziano Zito, and Laurenz Wiskott (2014). "An extension of slow feature analysis for nonlinear blind source separation". In: *The Journal of Machine Learning Research* 15.1, pp. 921–947.

Sreenivasan, Sameet and Ila Fiete (2011). "Grid cells generate an analog error-correcting code for singularly precise neural computation". In: *Nature neuroscience* 14.10, pp. 1330–1337.

Stachenfeld, Kimberly L, Matthew M Botvinick, and Samuel J Gershman (2017). "The hippocampus as a predictive map". In: *Nature neuroscience* 20.11, pp. 1643–1653.

Steffenach, Hill-Aina et al. (2005). "Spatial memory in the rat requires the dorsolateral band of the entorhinal cortex". In: *Neuron* 45.2, pp. 301–313.

Steinberg, Benjamin et al. (2016). *Representation theory of finite monoids*. Springer.

Stemmler, Martin, Alexander Mathis, and Andreas VM Herz (2015). "Connecting multiple spatial scales to decode the population activity of grid cells". In: *Science Advances* 1.11, e1500816.

Stensola, Hanne et al. (2012). "The entorhinal grid map is discretized". In: *Nature* 492.7427, pp. 72–78.

Stensola, Tor et al. (2015). "Shearing-induced asymmetry in entorhinal grid cells". In: *Nature* 518.7538, pp. 207–212.

Sterling, Peter and Simon Laughlin (2015). *Principles of neural design*. MIT press.

Stokes, Mark G et al. (2013). "Dynamic coding for cognitive control in prefrontal cortex". In: *Neuron* 78.2, pp. 364–375.

Strausfeld, Nicholas J et al. (1998). "Evolution, discovery, and interpretations of arthropod mushroom bodies". In: *Learning & memory* 5.1, pp. 11–37.

Stroud, Jake P et al. (2023). "Optimal information loading into working memory explains dynamic coding in the prefrontal cortex". In: *Proceedings of the National Academy of Sciences* 120.48, e2307991120.

Sun, Chen et al. (2020). "Hippocampal neurons represent events as transferable units of experience". In: *Nature neuroscience* 23.5, pp. 651–663.

Székely, Gábor J and Maria L Rizzo (2013). "Energy statistics: A class of statistics based on distances". In: *Journal of statistical planning and inference* 143.8, pp. 1249–1272.

Tanaka, Hirokazu (2016). "Modeling the motor cortex: Optimality, recurrent neural networks, and spatial dynamics". In: *Neuroscience research* 104, pp. 64–71.

Tang, Mufeng, Helen Barron, and Rafal Bogacz (2024). "Learning grid cells by predictive coding". In: *arXiv preprint arXiv:2410.01022*.

Tatli, Gokcan and Alper T Erdogan (2021a). "Generalized polytopic matrix factorization". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3235–3239.

— (2021b). "Polytopic matrix factorization: Determinant maximization based criterion and identifiability". In: *IEEE Transactions on Signal Processing* 69, pp. 5431–5447.

Taube, Jeffrey S (2007). "The head direction signal: origins and sensory-motor integration". In: *Annu. Rev. Neurosci.* 30.1, pp. 181–207.

Taube, Jeffrey S, Robert U Muller, and James B Ranck (1990a). "Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis". In: *Journal of Neuroscience* 10.2, pp. 420–435.

— (1990b). "Head-direction cells recorded from the postsubiculum in freely moving rats. II. Effects of environmental manipulations". In: *Journal of Neuroscience* 10.2, pp. 436–447.

Templeton, Adly et al. (2024). "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet". In: *Transformer Circuits Thread*. URL: `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

Tenenbaum, Joshua B and William T Freeman (2000). "Separating style and content with bilinear models". In: *Neural computation* 12.6, pp. 1247–1283.

Thompson, Emmett J et al. (2024). "Replay of procedural experience is independent of the hippocampus". In: *bioRxiv*, pp. 2024–06.

Tian, Zhenghe et al. (2024). "Mental programming of spatial sequences in working memory in the macaque frontal cortex". In: *Science* 385.6716, eadp6091.

Todorov, Emanuel, Athanassios Siapas, and David Somers (1996). "A model of recurrent interactions in primary visual cortex". In: *Advances in neural information processing systems* 9.

Tolman, Edward C (1948). "Cognitive maps in rats and men." In: *Psychological review* 55.4.

Touretzky, David S and A David Redish (1996). "Theory of rodent navigation based on interacting representations of space". In: *Hippocampus* 6.3, pp. 247–270.

Towse, Benjamin W et al. (2014). "Optimal configurations of spatial scale for grid cell firing under noise and uncertainty". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 369.1635, p. 20130290.

Tye, Kay M et al. (2024). "Mixed selectivity: Cellular computations for complexity". In: *Neuron*.

Van Cauter, Tiffany et al. (2013). "Distinct roles of medial and lateral entorhinal cortex in spatial cognition". In: *Cerebral Cortex* 23.2, pp. 451–459.

Varga, Adrienn G and Roy E Ritzmann (2016). "Cellular basis of head direction and contextual cues in the insect brain". In: *Current Biology* 26.14, pp. 1816–1828.

Ven, Gido M van de, Hava T Siegelmann, and Andreas S Tolias (2020). "Brain-inspired replay for continual learning with artificial neural networks". In: *Nature communications* 11.1, pp. 1–14.

Vollan, Abraham Z et al. (2025). "Left–right-alternating theta sweeps in entorhinal–hippocampal maps of space". In: *Nature*, pp. 1–11.

Waddell, Scott (2013). "Reinforcement signalling in Drosophila; dopamine does it all after all". In: *Current opinion in neurobiology* 23.3, pp. 324–329.

Walton, Mark E et al. (2010). "Separable learning systems in the macaque brain and the role of orbitofrontal cortex in contingent learning". In: *Neuron* 65.6, pp. 927–939.

Wang, Jane X et al. (2018). "Prefrontal cortex as a meta-reinforcement learning system". In: *Nature neuroscience* 21.6, pp. 860–868.

Wang, Mingye, Stefano Fusi, and Kim Stachenfeld (2025). "Brain-like slot representation for sequence working memory in recurrent neural networks". In: *Second Workshop on Representational Alignment at ICLR 2025*.

Wang, Shi-Qi et al. (2022). "Sexual selection promotes giraffoid head-neck evolution and ecological adaptation". In: *Science* 376.6597, eabl8316.

Waniek, Nicolai (2020). "Transition scale-spaces: A computational theory for the discretized entorhinal cortex". In: *Neural computation* 32.2, pp. 330–394.

Wanner, Adrian A and Rainer W Friedrich (2020). "Whitening of odor representations by the wiring diagram of the olfactory bulb". In: *Nature neuroscience* 23.3, pp. 433–442.

Wei, Xue-Xin, Jason Prentice, and Vijay Balasubramanian (2015). "A principle of economy predicts the functional architecture of grid cells". In: *Elife* 4, e08362.

Wen, John H et al. (2024). "One-shot entorhinal maps enable flexible navigation in novel environments". In: *Nature* 635.8040, pp. 943–950.

Wen, Quan and Dmitri B Chklovskii (2005). "Segregation of the brain into gray and white matter: a design minimizing conduction delays". In: *PLoS computational biology* 1.7, e78.

Whittington, James et al. (2018). "Generalisation of structural knowledge in the hippocampal-entorhinal system". In: *Advances in neural information processing systems* 31.

Whittington, James CR, Will Dorrell, et al. (2023). "Disentanglement with biological constraints: A theory of functional cell types". In: *The Eleventh International Conference on Learning Representations*.

Whittington, James CR, William Dorrell, et al. (2025). "A tale of two algorithms: Structured slots explain prefrontal sequence memory and are unified with hippocampal cognitive maps". In: *Neuron* 113.2, pp. 321–333.

Whittington, James CR, Timothy H Muller, et al. (2020). "The Tolman-Eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation". In: *Cell* 183.5, pp. 1249–1263.

Williams, Christopher KI (1998). "Computation with infinite neural networks". In: *Neural Computation* 10.5, pp. 1203–1216.

Wolpert, Daniel M, R Chris Miall, and Mitsuo Kawato (1998). "Internal models in the cerebellum". In: *Trends in cognitive sciences* 2.9, pp. 338–347.

Xie, Marjorie et al. (2022). "Task-dependent optimal representations for cerebellar learning". In: *bioRxiv*.

Xie, Xiaohui, Richard HR Hahnloser, and H Sebastian Seung (2002). "Double-ring network model of the head-direction system". In: *Physical Review E* 66.4, p. 041902.

Xie, Yang et al. (2022). "Geometry of sequence working memory in macaque prefrontal cortex". In: *Science* 375.6581, pp. 632–639.

Xu, Dehong et al. (2025). "On conformal isometry of grid cells: Learning distance-preserving position embedding". In: *The Thirteenth International Conference on Learning Representations*.

Xu, Yilun et al. (2020). "A Theory of Usable Information under Computational Constraints". In: *International Conference on Learning Representations*.

Yamins, Daniel LK et al. (2014). "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the national academy of sciences* 111.23, pp. 8619–8624.

Yang, Guangyu Robert et al. (2019). "Task representations in neural networks trained to perform many cognitive tasks". In: *Nature neuroscience* 22.2, pp. 297–306.

Yang, Xiaojiang et al. (2022). "Nonlinear ica using volume-preserving transformations". In: *International Conference on Learning Representations*.

Yartsev, Michael M and Nachum Ulanovsky (2013). "Representation of three-dimensional space in the hippocampus of flying bats". In: *Science* 340.6130, pp. 367–372.

Ying, Johnson et al. (2022). "Disruption of the grid cell network in a mouse model of early Alzheimer's disease". In: *Nature Communications* 13.1, p. 886.

Yoder, Ryan M and Jeffrey S Taube (2009). "Head direction cell activity in mice: robust directional signal depends on intact otolith organs". In: *Journal of Neuroscience* 29.4, pp. 1061–1076.

Yoon, KiJung et al. (2016). "Grid cell responses in 1D environments assessed as slices through a 2D lattice". In: *Neuron* 89.5, pp. 1086–1099.

Yosinski, Jason et al. (2014). "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems* 27.

Yu, Changmin, Timothy EJ Behrens, and Neil Burgess (2020). "Prediction and Generalisation over Directed Actions by Grid Cells". In: *arXiv preprint arXiv:2006.03355*.

Zavitz, Daniel et al. (2021). "Connectivity patterns that shape olfactory representation in a mushroom body network model". In: *bioRxiv*, pp. 2021–02.

Zeger, Emi and Mert Pilanci (2025). "Unveiling Hidden Convexity in Deep Learning: A Sparse Signal Processing Perspective". In.

Zenke, Freidemann (2019). *SpyTorch*. https://zenodo.org/record/3724018#.Yy7coOzML9t.

Zenke, Friedemann and Tim P Vogels (2021). "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks". In: *Neural computation* 33.4, pp. 899–925.

Zhang, Wenhao, Ying Nian Wu, and Si Wu (2022). "Translation-equivariant representation in recurrent networks with a continuous manifold of attractors". In: *Advances in Neural Information Processing Systems* 35, pp. 15770–15783.

Zhang, Zhewei et al. (2018). "A neural network model for the orbitofrontal cortex and task space acquisition during reinforcement learning". In: *PLOS Computational Biology* 14.1, e1005925.

Zheng, Zhihao, J Scott Lauritzen, et al. (2018). "A complete electron microscopy volume of the brain of adult Drosophila melanogaster". In: *Cell* 174.3, pp. 730–743.

Zheng, Zhihao, Feng Li, et al. (2022). "Structured sampling of olfactory input by the fly mushroom body". In: *Current Biology* 32.15, pp. 3334–3349.

Zhou, Jingfeng, Matthew PH Gardner, et al. (2019). "Rat orbitofrontal ensemble activity contains multiplexed but dissociable representations of value and task structure in an odor sequence task". In: *Current Biology* 29.6, pp. 897–907.

Zhou, Jingfeng, Chunying Jia, et al. (2021). "Evolving schema representations in orbitofrontal ensembles during learning". In: *Nature* 590.7847, pp. 606–611.

Zhou, Shanglin, Sotiris C Masmanidis, and Dean V Buonomano (2022). "Encoding time in neural dynamic regimes with distinct computational tradeoffs". In: *PLOS Computational Biology* 18.3, e1009271.

Zimnik, Andrew J and Mark M Churchland (2021). "Independent generation of sequence elements by motor cortex". In: *Nature neuroscience* 24.3, pp. 412–424.

Zipser, David and Richard A Andersen (1988). "A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons". In: *Nature* 331.6158, pp. 679–684.