What is the Neural Code?

Advanced Project COMSM3100

Adam Matheson 1757290

Supervisor: Dr Cian O'Donnell

University of BRISTOL

Department of Computer Science

Faculty of Engineering

September 7, 2018

Abstract

Whilst many areas of human ignorance still exist despite scientific advancement, the inner workings of the mind stand out as true frontiers which have hardly even been breached. The biomechanical processes by which information is represented in the brain are relatively well known - cells called neurons sending patterns of electrical signals to each other to communicate. But a key question in neuroscience is: why does a particular stimuli result in a particular pattern of activity in the brain? A large area of research is trying to understand this mapping between stimulus and response to answer questions like: what are the properties of this representation of the outside world? Why is information represented in this particular way? What is the brain doing to translate this visual input into a seemingly indecipherable series of billions of tiny electrical signals? This project attempts to detangle this 'neural code' using computational assistance, to better understand how the brain represents visual cortex activity in response to visual stimuli - taking one step further on the journey to devising a more complete understanding of how the brain works.

The project is primarily research-based (70% Type II); with some requirement to develop software to test the theory (30% Type I) using existing publicly-accessible data produced by experiments conducted by The Allen Institute for Brain Science. Some of the software already exists as accompaniments to existing research papers, and was assessed to decide whether it requires extension. The large quantity of data describing stimuli and brain responses favours machine learning methods, which are identified in an extensive literature review. These methods are used to attempt to retrieve the mapping from stimulus to response with the goal of producing models that describe the data. The project uses existing machine learning techniques to infer a model for the neural code in the context of the visual cortex, to initiate a discussion of the form and features of the neural code itself. The interpretability of the models undergoes extensive analysis to facilitate knowledge extraction. Fully understanding them will enable a more complete assessment of their suitability in the context of all possible models.

A set of implementations are constructed that successfully model the underlying data, enabling classification of unseen data, reproduction of probability distributions and sampling from learned models. It is found that performance varies based on a number of factors such as brain region. Probability distributions are used to compare these factors, as well as comparisons with baseline models.

This project delivers value through its contribution to the global effort to analyse neural data in an attempt to understand the inner workings of the brain. This manifests as quantitative analysis of the models alongside qualitative analysis of the neural code through model interpretation.

Keywords: *represented, stimulus, response, mapping, model, interpretability*

Acknowledgments

I would like to thank my supervisor Cian O'Donnell for his invaluable counsel, responses to many e-mails and for giving his time generously. Our meetings were not only useful for my project, but were interesting discussions which contributed to my appreciation of computational neuroscience and my course in general.

I appreciate the work of the faculty and the University of Bristol generally, for an excellent academic year which gave me the ability to carry out this project. I want to thank all my lecturers and the excellent facilities that have enabled this course.

A special mention goes to my coursemates, who have engaged with the programme in a way that has inspired me greatly. Our study sessions as general discussion about the subject enriched my knowledge of the field almost as much as the course content itself.

Most of all I would like to give my greatest thanks to my family and girlfriend for their constant support, without whom I would never have been able to reach this point.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Adam Matheson, September 7, 2018

Contents

Chapter 1

Introduction

1.1 Aims & Objectives

Information is represented in the brain as patterns of electrical activity in cells called neurons. A large area of research in neuroscience focuses on the 'neural code' - how the brain internally represents information such as sensory stimuli (Meister and Berry, 1999). Understanding the mapping between sensory stimuli and the brain's electrical activity has many challenges. The brain comprises ten of billions of neurons (Azevedo et al., 2009), meaning the scale of this information representation is enormous. Furthermore, research suggests that several layers of processing take place; neural coding seeks to understand how information is transformed as it moves through these layers, and whether some of it is discarded during the process (Richmond, 2009). Yet more research seeks to understand how correlations in patterns across populations of multiple neurons rather than single neurons influences the coding.

The Allen Institute for Brain Science is a Seattle-based non-profit bioscience research body working towards a better understanding of how the brain works. They have recently released a dataset representing physiological activity in the visual cortex of 199 mice (Allen Brain Institute, 2017a), (Allen Brain Institute, 2017b). This is the region of the brain that receives visual input from the retina in the eye. Over a number of sessions, visual stimuli such as images and video were presented to the mice, and the brain activity in response to these stimuli recorded using a variety of techniques (see section 1.5). The aim of this project is to decode this recorded activity to attempt to retrieve the initial stimulus, in order to infer a broader model of the neural code. Following this, a deeper investigation of this inference will comment further on the form of the neural code itself. Specific focus will be placed on whether there are discrete regions for coding specific stimuli, the impact of populations of multiple neurons coding for information, and how information behaves when interacting with multiple layers of processing.

The primary objectives are therefore:

1. Use existing machine learning (ML) code and implement further ML techniques if required, to analyse brain activity data from mice.
2. Utilise statistical tools to decode the stimulus and infer a model for the response. Example models include the the population tracking model (ODonnell et al., 2016) and pattern analysis techniques, among others.
3. Compare the efficacy of methods and models used.

4. Investigate the learned model to examine the form of the neural code itself.

1.2 Deliverables

Deliverables to satisfy the above objectives will be:

- Literature review of the state of the art in the field, to contextualise and focus the rest of the work in the project.
- Implementation of the techniques to learn the models, followed by quantitative analysis and comparison of various models' strengths and weaknesses.
- Qualitative and quantitative theoretical analysis drawing wider conclusions about the structure of the neural code, to produce some insight into the encoding for further work to build upon.

Together with the objectives in the previous section, these deliverables define the core metrics by which the success of the project will be measured. A framework for evaluation is essential from phase one of a project of this size; in order to prevent scope creep, missed deliverables, or missed schedules. Without overarching goals to aim for throughout the project, work will lack focus. Evaluation is ongoing and will be used to feedback into the project to adjust its direction regularly.

1.3 Added Value

The scale and complexity of the brain is such that data analysis must come from distributed multiple parties. Each piece of research mounts up, steadily improving the overview of the structure and workings of the brain; this project intends to play its part in that.

This dataset is very extensive in terms of number of imaging sessions, number of visual areas, and depths in the cortex. This enables analysis of larger populations of neurons to answer questions about how they interact, and whether encoding is done by singular neurons or by populations of interacting ones. As imaging technology improves and the number of neurons imaged simultaneously increases, techniques must be applied and adapted to these types of datasets to examine their scalability (Breakspear, 2017).

This project is therefore part of a greater march towards more comprehensive sampling and analysis of the brain. Whether the visual cortex as a region of the brain is structurally specialised to deal with visual stimuli, or if it is a more 'general purpose' region that coincidentally deals with visual stimuli, remains unclear. Should the case be the latter, the conclusions drawn from this research could generalise across the entire cortex, further increasing its value. All code will also be made publicly available (for example on GitHub), for further work to build upon.

1.4 Scope

The primary research area is the field of computational neuroscience, which explores the scientific discovery of the nervous system through techniques such as mathematical modeling and theoretical analysis, with computational aid. Primary areas of interest are brain structure, cognition and neural information processing.

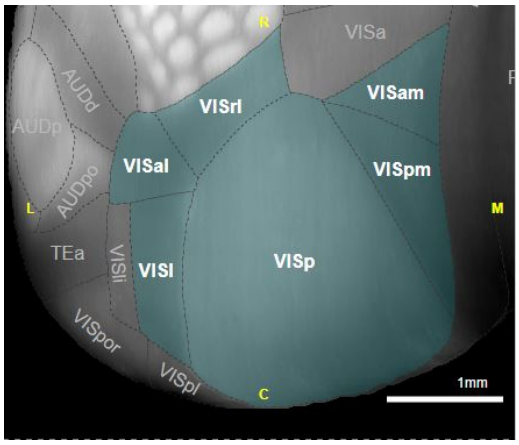


Figure 1.1: The mouse visual cortex involved in the Allen Intitute experiment in blue, subdivided into its critical imaging locations. *Source: (Allen Brain Institute, 2017b)*

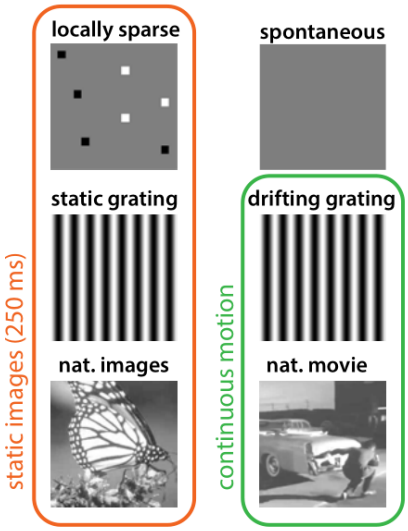


Figure 1.2: The categories of stimuli shown in the Allen Institute experiment. *Source: (Allen Brain Institute, 2017a)*

Machine learning (ML) has revolutionised data analysis across many fields. ML systems can learn from data, despite not being programmed to do so explicitly. They are excellent at extracting patterns, models and insights from datasets that would evade human analysis due to their size and complexity (Witten et al., 2016). This project will seek to apply existing ML methods to a new domain, then compare and contrast them.

In general, 'data science' (here meaning extracting insight and knowledge from data) describes the domain of the project. Techniques within data science besides ML, such as pre-processing and visualisation, may be utilised to varying degrees.

1.5 Materials

Since the project is primarily concerned with data analysis, it is important to be familiar with the data in question early on. Data was acquired from the Allen Brain Observatory, the initiative within the Allen Institute dedicated to conducting experiments examining physiological activity in the mouse visual cortex. The mice's genomes were altered through the use of genetic engineering ('transgenic' mice) which resulted in the presence of the GCaMP protein, specifically GCaMP6 - known as a 'calcium indicator'. When activity takes place in a neuron, calcium ions (Ca^{2+}) undergo a binding process and through the presence of GCaMP6, they fluoresce.

Transgenic mice were head-fixed and shown a variety of visual stimuli (see figure 1.1) at 512 x 512 pixel resolution, whilst in-vivo calcium (Ca^{2+}) imaging data was recorded for a varying number of neurons, to record response activity. The stimuli were presented across sessions A, B and C/C2 which were spread across three days. Further details for the experiments can be found in (Allen Brain Institute, 2017b).

Part I

Background

Chapter 2

Neuroscience Background

2.1 Neuroscience

The brain is made up of cells called neurons, which themselves comprise three main parts: the body (soma) , dendrites, and an axon (see figure 2.1). The neuron's purpose is to receive incoming electrical signals, and depending on certain characteristics of that signal, send it onwards to parts of the body; for example other neurons or muscles to trigger human behaviour such as thought or movement. In the majority of cases, signals are are received via the dendrites and transmitted via the axon. One neuron can have many dendrites and receive multiple inputs, such as the Purkinje neuron which is found in the cortex of the cerebellum - the brain region responsible for motor function. This type of neuron can have 200,000 dendrites (see figure 2.2). Each neuron outputs signal via the single axon, which can propagate a signal rapidly over long distances.

Neurons are connected to each other in vast networks via connections called synapses, and send and receive electrical signals through these connections. Upon receiving an incoming signal strength above some threshold, the neuron initiates a change in membrane potential (the charge of the neuron). This leads to the neuron emitting this potential as an electrical pulse called an 'action potential' (AP), otherwise known as a 'spike' (see figure 2.3). This pulse travels down the axon into other neuron's dendrites, subsequently causing those neurons to fire if the threshold is high enough, in a repeating pattern. Signals can either be excitatory (causing the neuron to spike) or inhibitory (causing a decrease in spiking rate). Following a spike trigger, neurons goes through a refractory period, which is a short length of time where no further spike can be triggered. As long as the neuron is not in this refractory period, it can fire multiple times in response to a stimulus to produce a series of spikes called a 'spike train'. This is the most common form of neural activity rather than isolated spikes (Brown et al., 2004).

In the brain there are around 85 billion neurons (Azevedo et al., 2009), resulting in roughly 100 trillion synapses. Therefore information is represented as different states of neural firing within the brain across this massive set of cells and connections at any time, initially observed by Adrian and Zotterman in 1926 when they noticed nerves emitting more spikes when the weight hung from a muscle increased (Adrian and Zotterman, 1926).

2.1.1 Brain Imaging

Crucial to projects that undertake large-scale data analysis is how the data is actually gathered. A range of techniques exist to record (image) neural activity in the brain. Two

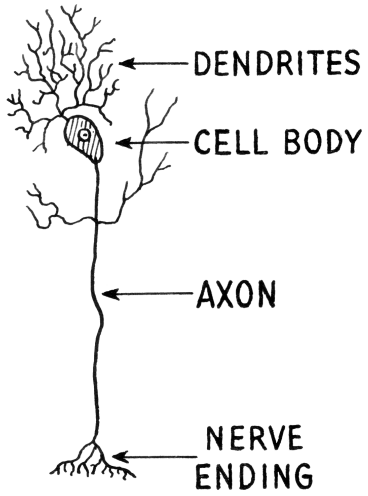


Figure 2.1: The typical structure of a neuron. *Source: Pearson Scott Foresman (2007). Retrieved from* `https://commons.wikimedia.org/wiki/File:Dendrite_(PSF).png`

Figure 2.2: A Purkinje cell in the cerebellar cortex, with a large number of dendrites (c) but still only one axon (a). *Source: Santiago Ramon y Cajal (1899). "Purkinje neurons from the human cerebellum". Cajal Institute, Madrid.*

simple in-vivo approaches are the sharp electrode method and the patch-clamp technique. For the former, a cell is punctured by an electrode to enable it to measure voltage inside the neuron (Gross et al., 1977). The patch-clamp technique improves on this by instead 'sucking' a small piece of the cell membrane into the electrode, forming a seal to enable intracellular recording (Hamill et al., 1981). Another technique is functional magnetic resonance imaging (fMRI). This technique records changes in blood flow, volume, and oxygenation in brain cells - exploiting the fact that as activity increases, so does blood flow (Logothetis et al., 2001). This has a fairly low spatial resolution of three to six millimeters, and a medium temporal resolution in the order of seconds; but has the significant advantage that it can be performed externally. Therefore it is extremely useful for scenarios where less accurate measurements are required and intrusiveness is a consideration; for example medical diagnoses in humans at a whole-brain scale, rather than accurate data for individual neurons.

A fourth type of imaging technique is calcium (Ca^{2+}) imaging, which was the technique used to record the data in the Allen Institute dataset. Proteins that increase neuron fluorescence (light emissions) when bound to Ca^{2+} inside cells are introduced via genetic modification; this fluorescence is then imaged during brain activity. This imaging technique has a high signal-to-noise ratio as it eliminates autofluorescence (natural emission of light by biological structures upon absorbing light), enabling it to image structures deep within tissue; making it perfect for this study where accurate measurements are required and regions of the visual cortex can be deep. One limitation is that it can suffer from low spatial resolution in certain situations (Russell, 2011). It also only produces a recording of the fluorescence, which doesn't perfectly match with actual brain activity because fluorescence changes much more slowly than the firing rate of neurons. Therefore, to extract neural activity from a fluorescence trace, an additional step must take place known as spike inference. The rate of change of fluorescence dF/F is calculated, and activity

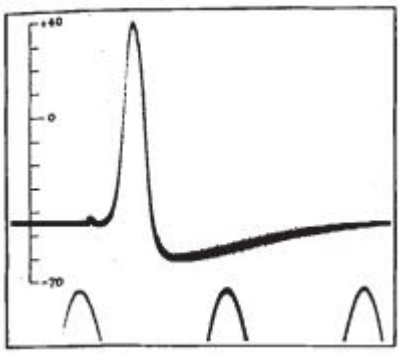


Figure 2.3: The typical structure of a neuron action potential, first recorded in-vivo via electrode by Hodgkin & Huxley in 1939. The time markers on the x-axis represent 500ms intervals. The y-axis indicates the potential in mV of the electrode. The neuron charges, reaches a threshold, spikes, then hyperpolarises back down to a rest state, including a short refractory period. *Source: Hodgkin and Huxley (1939)*

inferred from this metric, for which a range of techniques exist (Lewicki, 1998), (Berens et al., 2018).

2.2 Neural Coding

Overall, the biomechanical processes of information representation at a neuronal level as described previously in this chapter are well known, and a range of techniques to record them has been described. However, if patterns of firing neurons is indeed an accurate explanation of how information is represented in the brain, how does the brain manage to represent such a vast range of information simply through a set of neurons being either 'on' or 'off'? This is the central question dealt with by 'neural coding', a subfield of neuroscience.

Neural coding attempts to unravel some of the mystery in information representation by using models to map stimuli to responses in the brain (Dayan and Abbott, 2001). These responses are in the form of neurons firing to each other to communicate. If a large amount of data describing the relationship between spike patterns and a known stimulus is available, a model for that mapping can be constructed using inferential statistics.

There are many factors to consider when forming such a model. It is rarely as simple as decoding information from a set of spike trains, as was the case with the early research mentioned above (Adrian and Zotterman, 1926); and single-neuron studies have inherent flaws. Firstly, their propensity for bias. It is easy to target a neuron that demonstrates the behaviour the researcher wants to show, whilst ignoring others - this is difficult to achieve with neuron populations. Furthermore, as research progressed it became clear that the inferences made from individual neurons are not transferable when considering information encoded by a population of neurons. In fact, some studies have been found that models can be constructed that accurately predict the activity of single neurons from the state of all others; regardless of how well or poorly the single neuron codes for a specific piece of information (Fuhs and Touretzky, 2006), (Meshulam et al., 2017). Schneidman et al. (2006) also suggests strong relationships between groups of at least ten neurons, implying that the neural code has associative or error-correcting properties. The correlation between individuals encodes as significant amount of information (if not more significant) as the individuals themselves.

Another factor is the delay between stimulus and spike, as shown in a study by Izhikevich (2004). The study found that nuances in conduction delays can "give rise to the spontaneous formation of neuronal groups - sets of strongly connected neurons ... repeatedly generated patterns of activity with millisecond spike-timing precision". Furthermore, brain response changes depending on many characteristics of the stimulus: for visual stimuli these are intensity, motion, and length of exposure. It is also known that there is a spatial element to neural coding, with different brain regions responding to specific stimuli but remaining dormant for others. This may be even more subtle than first thought, with detailed specialisation for certain stimuli. For example, Schacter et al. (1995) identified a specialised brain system just for representing three dimensional structure. So many nuanced considerations mean neural coding is a difficult problem to tackle.

A further central issue that makes neural coding extremely difficult is the concept of neuronal variability. Neurons fire with high variability even in multiple trials of the same experiment when controlling for variables, as well as when in a rest state not exposed to a stimulus (including sleep) (Werner and Mountcastle, 1963). This is often regarded as noise; an artifact of the process of producing electrical impulses in the neuron. Taken as this alone, the introduction of a large amount of noise to a coding between stimulus and response makes the act of decoding that mapping very difficult. But to further complicate the problem, some research now points towards this variability as an integral component of information representation, rather than an artifact to be ignored - itself encoding some information (Stein et al., 2005). A theory proposed by Li and Tsien (2017) called neural self-information theory attempts to characterise neuronal variation across a spike train and how this variation may contribute information rather than simply exist as noise. Self-information is a principle in information theory, measuring the deviation from the expectation of a random variable when sampling from a set of random variables. The authors propose that the neural code is a self-information process based on the information carried not in spiking neurons themselves - but in the intervals between spikes. The implication of this theory is that through a model outlined in the paper, it enables decoding to find neural states from 'noise' alone.

Amongst all these intricate considerations however, by far the biggest issue is scale. Assuming individuals in a population of N neurons have two states, active or silent (on or off), this means the total number of possible patterns is 2^N. For just 20 neurons of the billions present in the brain, this number already equates to around 1 million possible patterns. For 128 neurons, a number commonly features in this experiment, the number of possible patterns is several orders of magnitude greater than the age of the universe (Riess et al., 1998). Our models must therefore aim to reduce the number of parameters, which will inherently also result in a reduction in the amount of information the model captures when compared with the true distribution. We should endeavor to build models with can reproduce probability distributions as closely as possible to what is believed to be, and what the data states is, the true distribution. The bigger the discrepancy between the modeled distribution and the true distribution, the less accurate the model will be.

This project deals with visual stimuli. In this context, an image or video is an incredibly rich repository of information. The brain views even a simple image as not just a collection of colour or shapes, but as a more sophisticated representation of a set of parts and relationships (Biederman, 1987). For example in an image of a car, it is understood that the image will contain wheels, doors, windows, and a body relating those components together into one object. This implies the perception of an image arises out of a constructive process (Stephen Michael Kosslyn, 1975). To simplify this and isolate responses to stimuli, it is useful to reduce a visual stimulus to simple parts such as basic shapes and patterns at different intensities and rotations. One approach taken by the data under consideration in

this project is that simple static and drifting grating patterns were used (see figure 1.2).

To formalise these considerations, it can be stated that a full characterisation of the stimulus-response mapping is contained in the conditional probability distribution over observed neuron states, given the sensory input. However for even moderately sized neuron populations, sampling of these distributions is intractable. The domain benefits from more powerful methods and models to fully understand the problems.

2.3 Conclusion

The basic domain knowledge required to fully appreciate the project and give the subject a comprehensive computational treatment have been briefly introduced, with further reading highlighted. One of the most central concepts, neural action potentials or 'spikes' were introduced, which in series called spike trains will comprise the core of our dataset. The key considerations when working with neural data were also highlighted in this chapter, namely: modeling populations of neurons rather than individuals; delays between stimuli/response; neuronal variability/noise; and most of all, scale. Chapter 3 will introduce broad frameworks designed to alleviate some of these problems, and chapter 4 will introduce some specific techniques in use today.

Chapter 3

Computational Methods

3.1 Computational Neuroscience

The field of computational neuroscience extends traditional neuroscience by utilising mathematical and computational techniques to gain insight into the structure, workings and function of the brain. In a symbiotic fashion, computational research continues to uncover how biological processes within the brain can often be expressed in a mathematical fashion, revealing how the brain can be viewed as an algorithmic unit itself.

The complexity of the brain is such that much of what neuroscience seeks to reveal remains obscured. In a summary of the goals of the US BRAIN initiative (a multi-billion dollar research program backed by the US government, developing understanding of the human brain), Jorgenson et al. (2015) states that neuroscience has yet to "mature to the extent that we discover basic principles of neural coding and computation". They draw attention to the exciting 'big data' that the advent of computational neuroscience has provided, but acknowledge the requirement to uphold rigorous analysis and theoretical insight to extract knowledge from that data.

Despite the remaining mystery, the field has made stunning breakthroughs recently. Deep neural networks, inspired by and abstracted from neuron structures in the brain, has solved many sophisticated problems such as image recognition (LeCun et al., 2015). Indeed, the brain can be viewed as being built of many recurrent versions of neural networks and Barak (2017) highlights recurrent neural networks' place in current neuroscience research. Strong evidence of this also comes from Yamins et al. (2014), in the context of convolutional neural networks. In this study, computational techniques were used to learn a neural network architecture that matches human performance in the domain of object classification. Upon further analysis it became clear that the model was highly predictive of neural responses in certain areas of the visual cortex, implying an optimisation process helped shape neural mechanisms like this one in biology. This example also illustrates the symbiotic relationship between biology and computational neuroscience; as biology helps build templates for computational systems, so do computers reveal more about the biology behind the organic systems.

3.2 Machine Learning

Computational neuroscience enables analysis of extremely large data sets to provide deeper insight. It is expected that purely due to increased scale, these large data sets will contain

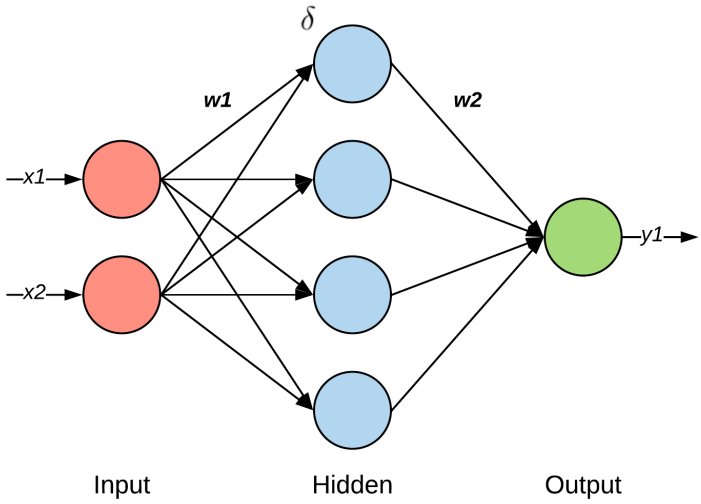


Figure 3.1: *A typical neural network architecture (called a 'multi-layer perceptron' (MLP)), as inspired by structures in the biological brain. Each node has a type of function for processing, usually sigmoidal (as inspired by a biological neuron). The input* $(x1, x2)$ *is weighted and fed forward to all nodes in the hidden layer. These nodes are biased* (δ) *to allow shifting of the activation function. If the input surpasses some threshold, it is fed forward again (similar to a neuronal spike). Eventually it reaches the output layer* $(y1)$*. The weights can be learnt to give the correct output for given inputs.*

more valuable knowledge than traditional smaller data sets; however, by definition they also include more useless data. These large data sets therefore come with their own problems - how to analyse them effectively and efficiently in order to extract knowledge and insight from the raw data.

ML enhances the human activity of learning that we have been engaging in for millenia with greatly increased speed and capacity. Essentially, ML identifies patterns in data, then takes action to exploit those patterns (Bishop, 2006). It can provide techniques to extract knowledge that may otherwise go unnoticed. These techniques are not explicitly programmed to solve individual problems, but are designed to deal with a range of problems within a specific domain - adapting and improving their solutions as the data available and feedback from iterations increases, in a form of learning. Therefore, an ML approach is perfect for this project's dataset, due to its large scale, and because its properties are largely unknown which therefore requires adaptive analytical techniques. The mathematics behind such techniques have often existed for a long time, but the advent of massive data sets and exponentially more powerful computers have made these approaches practical.

There are three main categories of ML: supervised learning, unsupervised learning, and reinforcement learning (with a fourth hybrid category: semi-supervised learning) (see figure 3.2). In the case of supervised learning, both a set of inputs (independent variables) and corresponding outputs (dependent variables) are known, constituting a training data set. The goal is to learn a function that maps these inputs to their outputs. The algorithm begins at random, with a low mapping accuracy, and utilising a learning function such as gradient descent it iterates over the data until a desired accuracy is reached. Following this, test data is used to verify the accuracy of the algorithm. Finally, new input data can be fed through the learned function to produce a (hopefully) accurate output. Examples of

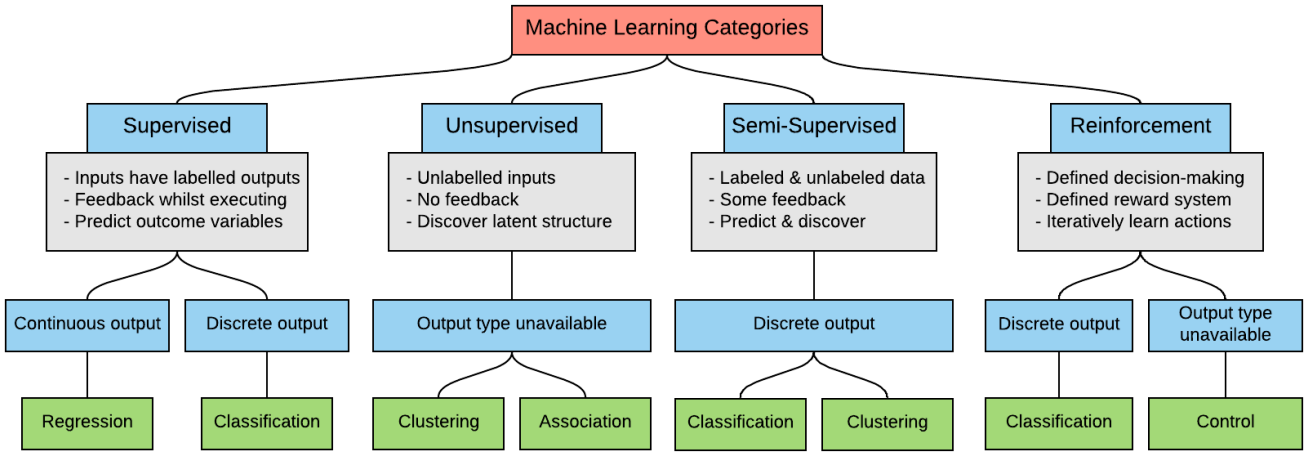


Figure 3.2: *An overview of the three main categories of Machine Learning algorithms.*

supervised learning techniques are regression and K-Nearest Neighbour. For unsupervised learning, input data remains but no target variable exists. These algorithms therefore have no explicit goal but can uncover characteristics of the data in order to cluster the population into groups. Examples include k-means clustering and mixture models. Therefore, the key difference between supervised and unsupervised learning is that the former is trained on labeled data to guide the algorithm into identifying features; the latter must allocate the important features by itself. The hybrid category mentioned, semi-supervised learning, works with both labeled and unlabeled data, to provide the benefits of supervised learning to domains where fully labeled data is impossible to access. Finally, in the case of reinforcement learning (RL), a system trains itself using trial and error. Making decisions based on a pre-defined decision process, the system is rewarded according to a specified reward function for a good outcome and can be punished for a bad outcome, thus tending towards the best knowledge representation. Examples of RL are Q-Learning and Markov decision processes.

3.2.1 Machine Learning in Computational Neuroscience

These techniques can be applied to neural coding. Supervised learning is most appropriate for this project, as both inputs and corresponding outputs are available in the Allen Institute data set. If a model can be specified over this data, it could be sampled from to try and predict the brain response when a new stimulus is presented (see figure 3.3 for a visual description of this process). Moreover, if this model can be generalised across other areas of the brain, it could provide great insight for researchers in other fields of neuroscience.

There are some unique features of this domain that should be considered when learning a model. The data is likely to be sparse due to the large number of neurons in the brain. If there is an element of spatial encoding, then it follows that for any individual stimulus, for the group of neurons that fire, there will be a large number of neurons that do not fire in the rest of the brain. Whilst proving this would require monitoring of a vast number of neurons which is not yet possible, current research points towards this being the case. For example, Ganmor et al. (2011) show that a "very sparse low-order interaction network underlies the code of large populations of neurons". The paper describes a model framework that would take advantage of this knowledge to increase efficiency.

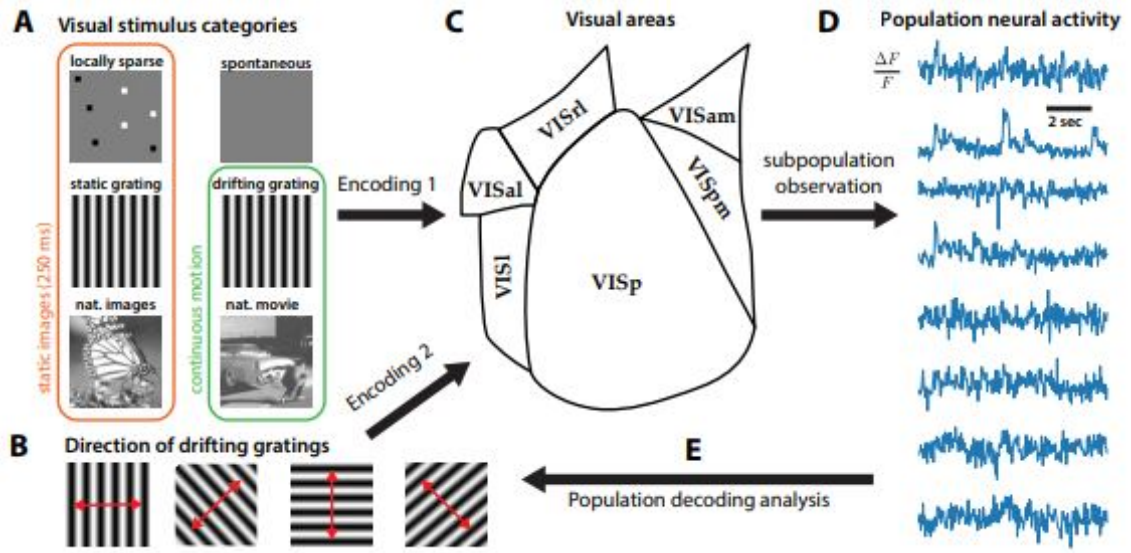


Figure 3.3: *The stages involved in learning a model for the Allen Institute data in this project.* ***A:*** *Six visual categories and* ***B:*** *eight directions of static/drifting grating stimulus are shown.* ***C:*** *Excitatory neuron activity in the mouse visual cortex is analysed.* ***D:*** *Normalised signals are recorded.* ***E:*** *A model is learnt from known decoding data, to produce a model for decoding future unseen stimuli. Source: (Esfahany et al., 2017)*

3.2.2 A Note on Interpretability

Another key consideration for this domain, which is often overlooked in other applications of machine learning, is interpretability (Doshi-Velez and Kim, 2017). Computational neuroscientists are striving to find answers to complicated questions and explanations for natural phenomena; whereas in a commercial setting a machine learning algorithm may simply be employed to boost profits or increase efficiency. In this latter case the outcome is the primary consideration, but in the former, the process of understanding how the system has reached its outcome is equally, if not more important. Whilst it is possible to assess the accuracy of a black box by outcome alone, one cannot comment on that accuracy in the context of other models. If the researcher cannot identify whether or why the model is the best among all possible models, its usefulness is greatly diminished. Therefore a key aspect of this project will be interpreting the model after it has been learned.

3.3 Conclusion

This chapter draws the fields of neuroscience and computer science together into computational neuroscience, explaining how cutting edge statistical and data analytic techniques can be leveraged to expedite neural coding research. The various types of ML techniques were assessed and compared, with comments on their applicability to various types of dataset. It was decided that the characteristics of the Allen Institute dataset match well with unsupervised learning methods due to the availability of both input data and labeled outputs. Some specific features of the dataset which may need to be considered in the project execution were also highlighted. Finally, interpretability was highlighted as a key consideration to achieve the goals set out in chapter 1.

Chapter 4

ML Models for Neural Population Data

ML systems have been applied to neural coding for some time, and as researchers realised that in some cases neural coding can be treated similarly to other data pattern analysis problems, uptake increased further (Pereira et al., 2009). Work is ongoing to attempt to detangle and model many of the complexities of the neural code identified in the previous chapters.

Broadly, there are two classes of model. Firstly, those that model neural activity accounting for internal dynamics only. Often, these aim to predict the probability of spiking activity across a certain number of neurons, conditioned on various dynamics of the system such as the state of other neurons. This prediction can be for anything from single neurons, pairwise correlations, higher-order correlations, all the way up to the overall number of spiking neurons (an important metric, known as the 'population rate' and described by K). The second class of model is those that model neural response to a given stimulus.

Additionally, some models deal only with synchronous activity (neural population state at a single point in time), whereas others condition on time-based aspects of data to incorporate temporal correlations across time bins.

4.1 Internal Dynamics

Models within the maximum entropy (ME) framework are a popular approach when modeling response probabilities. This class of model strips assumptions and gives the least biased estimate possible for a probability distribution, given the available information (Jaynes, 1957). This information is an incomplete representation of the true distribution, so takes the form of a set of constraints on the space of total possible probability distributions (Uffink, 1997). This takes the form

$$S(P) - \sum_{\mathbf{x}} P(\mathbf{x}) \log P(\mathbf{x})$$

... where $\mathbf{x}$ contains the neural responses. This produces the most random distribution that still models the underlying system. The most simple example of this is using the ME framework to reproduce the population rate K (Tkaik et al., 2013):

$$P(\mathbf{x}) \propto \exp\left[g(K(\mathbf{x}))\right])$$

... where K once again denotes the population rate. This is trivial to learn, simply counting the number of active neurons at any given point (a process known as 'histogramming'). Another simple example is the model of independent neurons where the mean activity of neurons is constrained - a simple way of generating data and a good introductory example, but in reality not useful for modeling neuronal populations, as neurons are assumed independent and therefore correlations do not exist.

ODonnell et al. (2016) work towards improving understanding of large population neural coding by developing a new technique for analysis of the large quantity of spiking data generated on this scale. The contribution of a simple, scalable statistical technique called the population tracking model to characterise activity in large populations combats the issue of having to process 2^N connections (where N is the number of neurons recorded), instead resulting in a model of complexity N^2. The probability of seeing any pattern $p(x)$ is written as:

$$p(\{x\}) = \frac{p(K)}{a_k} \left(\prod_{i=1}^{N} p(x_i|K)^{x_i} \left[1 - p(x_i|K)\right]^{1-x_i} \right)$$

... where a_k is a normalisation constant. This constant is the sum of the probabilities of all patterns of an independent model of neurons, used to ensure the final probabilities sum to 1:

$$a_K = \sum_{\{x\} \in S(K)} \left(\prod_{i=1}^{N} p(x_i|K)^{x_i} \left[1 - p(x_i|K)\right]^{1-x_i} \right)$$

So, given the constraints of specifying the distributions for the population count $P(K)$ and the neuron activity given that population count $P(xi|K)$, we are left with a set of possible distributions denoted by $S(K)$. We simply ignore all distributions where $\sum_{i=1}^{N} x_i \neq K$, resulting in a set of distributions that are valid. These are combined and normalised using a_K as described above, to complete the reproduced distribution. This model is related to ME concepts and provides a good balance between accuracy and cost. The authors point out that one limitation of this model is its inability to account for temporal correlations. The technique is extremely promising and enables working with large values of N, although questions remain regarding how far the model can scale. Since it models probability distributions of individual neurons and of K, there comes a point with increasing N where the gap between these two modeled aspects will be large enough that there will be middle-order statistical correlations not being captured effectively by the model. Additionally, interpretability once again comes under scrutiny, as at this time despite the low number of parameters, the parameter values themselves are not interpretable. The model is similar to the population coupling model (Okun et al., 2015), but the population tracking model provides a distinct technique with which one can sample directly from model distributions - this is difficult to achieve with the population coupling model. The authors also claim that this model provides a low-dimensional model of the entire distribution, which the coupling model fails to achieve. Furthermore, the population tracking model is fully probabilistic and handles uncertainty, which should be considered an advantage because biologically the brain can be viewed as a machine to deal with uncertainty in perceptions. Knill and Pouget (2004) provides a Bayesian coding hypothesis arguing for a Bayesian approach to the neural code by collecting a body of work in psychophysics that points to perceptual computations in the brain being 'Bayes optimal' - meaning they reduce error in a probabilistic manner. However, it should be noted that during decision-making humans often view the possible outcomes in a very discrete manner, and are often able to come to decisions quickly and definitively without necessarily being able to concretely explain why; which leads to questions about how applicable a probabilistic treatment of neural coding at a biological level really is.

Tkaik et al. (2014) provide another sophisticated approach within the ME framework. The structure of their model is such that for each observation, only the minimum amount of structure required is added to the model to reproduce the response to that observation. They go into detail regarding the number of neurons they believe to be significant when considering relationships: for single neurons, interactions can be considered as negligible perturbations; for populations of around ten neurons it becomes detrimental to ignore these interactions and pairwise correlations should be considered; and for fourty or more neurons, even pairwise relations are not enough - even they must be augmented by global interactions controlling distributions of synchronicity. To formalise this concept, once can say the distribution of possible states has some structure that cannot be factorised into groups of neurons, let alone individuals. Indeed, using the model in the paper, the authors find the network's space of states has "considerable structure", and that the "probability of the same state repeating is many orders of magnitude larger than expected for independent neurons ... this is really quite startling" (p. 18). It should be considered that a limitation of maximum entropy models is that they can only ever state upper bounds on total and noise entropy, rather than define a strict bound - thus their reading remains just an estimate of information; albeit an accurate one under the right circumstances.

The ME approach can be extended yet further by constructing a model capable of modeling the firing rates of all neurons, and the correlation between all pairs of neurons. This extension is known as the Ising model and is exactly what the model in Meshulam et al. (2017) achieves:

$$P(\mathbf{x}) \propto \exp\left(\sum_i h_i x_i + \sum_{j \neq i} J_{ij} x_i x_j\right) = \exp(\mathbf{h^T x} + \mathbf{x^T J x})$$

... where $\mathbf{h}$ is a vector of coefficients and $\mathbf{J}$ is a matrix of interaction couplings (making it of similar form to the Hopfield model). It was found that the model described above accurately predicts the activity for a selected neuron, given the state of all other neurons - without any further assumptions, in accordance with ME principles. This produces a strong suggestion that effective models should carefully consider internal network state alongside or even instead of external factors such as stimuli.

This model can however break down and fail to capture higher-order correlations for large N (Tkaik et al., 2006). Steps taken to mitigate this issue include combining Ising models to build up more complex models, for example in the case of hierarchical models (Santos et al., 2010) - where neurons are pooled with each pool being modeled using the Ising approach, and interactions between pools rather than between individual neurons taking precedence; and semiparametric energy-based models - a generalisation of the Ising approach which learns a slightly different model structure from data, combating errors in the default Ising architecture (Humplik and Tkaik, 2016). Another way of attempting to capture higher-order correlations is to simply force the population rate K to be one of the observables constrained by the ME approach, known as a K-pairwise model (noting that K here refers to the population rate, not the commonly encountered use of k to represent divisions such as in k-means clustering). Both of these approaches improved the performance of the Ising model in their respective papers.

Following this hierarchical approach attempting to cover all orders of statistical correlations in a model to its natural conclusion, we arrive at the full log linear model (Martignon et al., 1995), which captures all distributions.

$$P(\mathbf{x}) \propto \exp\left(\sum_i \Theta_i x_i + \sum_{i_1 i_2} \Theta_{i1 i2} x_{i1} x_{i2} + \ldots + \Theta_{1 \ldots N} x_{1 \ldots N}\right)$$

... where neuron interactions are set by coefficients Θ. A value of 0 means no interaction, so clearly this defines a model hierarchy of infinite order, with all orders beyond that of the model in question simply having their coefficient set to 0. Obviously since it includes all distributions, it has up to 2^N parameters - intractable for any meaningful value on N but useful theoretically. Whilst all models discussed can be interpreted as restricting the full log linear form, the reliable interaction model in particular can be seen as a restriction that assumes no maximum order, whilst not having to learn all low-order correlations Ganmor et al. (2011). It does this by defaulting coefficients above to 0 for all but the most common observed responses, tackling the issue of sampling from a full distribution of neuron states. Initially they experiment with treating each neuron as independent, which results in massive errors when predicting network synchrony (simultaneous activity in populations of connected neurons). When looking at data, a number of patterns were identified, both spatial (positional patterns) and spatiotemporal (positional and time-related patterns). This relates back to the identification of sparsity earlier in this review - these patterns exist because of the sparsity of data. As less valued data points exist, it becomes more likely that they are likely to display patterns. These patterns can be used as a kind of heuristic to more efficiently estimate the joint activity interactions, and thus build a model that attempts to give the distribution of neural responses to various stimuli. In the example in the paper, this resulted in far fewer parameters than even a maximum entropy model - 450 versus 5,000 respectively, for $N = 99$. These interaction patterns suggest the broader neural code is learnable from interactions alone.

The Restricted Boltzmann Machine (RBM) is another class of algorithm roughly falling within the ME framework, its primary benefit being the capability to model higher-order correlations. Modeling these are usually quite difficult as the number of parameters quickly increases with N, but RBMs achieve this through use of latent variables - through which neurons are indirectly correlated. This joint probability between latent variables and neurons is denotes by:

$$P(\mathbf{x}, \mathbf{y}) \propto \exp(\mathbf{a^T x} + \mathbf{b^T y} + \mathbf{y^T w x})$$

... where y are the hidden variables, and a and b are vectors controlling the activity of neurons and y. This approach is explored in relation to the neural code in a paper by (Kster et al., 2014). In this particular study, a standard Boltzmann Machine (BM) was hybridised with an Ising Model by adding hidden units. In this example, both fully and semi-restricted (SR) BM were formed - SRBM being a variant with relaxed criteria allowing interactions between neurons. These hidden units introduce latent variables which enabled the model to capture state distributions in the network significantly better, and when combined with temporal data it can predict future network states based on the past. The prediction accuracy was significantly better than Ising models alone (between 10% and 40% depending on the size of the network), and outperforms maximum entropy models. The authors highlight that ME models suffer from the restriction of approximation (mentioned above) and that despite a reduction in the number of parameters in ME approaches, this number still grows rapidly with population size - both of which are mitigated with the RBM by using minimum probability flow (MPF) to estimate parameters efficiently. However, interpretability in RBMs is very poor, as is inherent when utlising latent variables as by definition they are not directly observed.

Models accounting for internal dynamics can also include a temporal element with minimal effort. In fact, due to their structure, neurons in different time bins can be effectively considered as entirely different neurons. This enhancement of the models above are called 'spatio-temporal' variants and are simple to construct, however they cannot describe joint activity probability distributions that do not change when temporally shifted. Other time-

based enhancement include the temporal RBM, and adapting ME models reproducing K to focus on the shifting properties of K itself rather than changing individual neurons.

Finally, linear dynamical systems Gao et al. (2016) attempt to adapt the latent variable modeling concept from RBMs to include temporal elements. The latent variables play the same role as in RBMs where neurons are correlated through them indirectly, but in linear dynamical systems, these latent variables are also correlated across time bins - therefore, so are neurons (indirectly). In this particular study, the system was able to predict hand-movements from neural activities changing over time in the macaque motor cortex with good accuracy.

4.2 Stimulus

The simplest way of modeling neural response when a labeled stimulus is involved is perhaps the Peristimulus Time Histogram (PSTH) method (Kster et al., 2014). The mean fluorescence is computed for each time bin, after taking repeated measures of responses. Then, an ME model is constructed to reproduce these means and neuron correlations.

$$p(\mathbf{x_t}) \propto \exp(\mathbf{h_t^T x} + \mathbf{x^T J x})$$

... where $\mathbf{h}$ models fields across time bins (stimulus-dependent), with $\mathbf{J}$ modeling the couplings. This approach is very easy to learn, however it is limited as it does not generalise to new stimuli - although is used as a basis for other models.

Far more intriguing are a class of model called transition models (Pillow et al., 2005). These are interesting because they incorporate mechanics that try to model both internal dynamics, *and* response to stimuli. The particular avenue of achieving this is by conditioning on both the activity in earlier time bins as well as the stimulus. They are also interesting because they are more clearly rooted in biological processes, modeling a current driving action potentials in neurons, as described in chapter 2. One variety of transition model is the generalised linear model, which includes temporal correlations. It models the firing rate λ_i depending on the input current I (hence the similarities to biological processes):

$$\lambda_{it} = f_i(I_{it})$$

... where f_i is an exponential rate function:

$$f_i(I) = \exp(\mu_i + I)$$

... where μ_i controls firing rate if there is no I. Research from Schwartz et al. (2012) explores neural coding for spatial information in retinal ganglion cells. The study monitored activity in 162 neurons when exposed to flashed images and compares the performance of linear and nonlinear decoders when retrieving the mapping. The complexity of the decoders varied to account for between one and five spikes and various time bins (discrete divisions of the continuous time period). The researchers conclude that linear decoders enable coarse categorisation of the shape stimuli, but subtleties in the stimuli are only captured by nonlinear decoders that consider correlation and spike timing. Their findings also imply a rough hierarchy for representation of visual information (layers), although as the population sizes increased the accuracy of the discrimination diminished. This implies that further accuracy may not be obtained using these systems even with substantially larger neural populations, limiting the usefulness of these methods. The study also falls short of suggesting an optimal complexity for the representations and simply comments that "more complex representations allowed significantly lower error rates", which seems

obvious. The authors then go on to compare the performance of an enhanced decoder, "based only on the pattern of pairwise correlation among neurons, rather than requiring knowledge of the stimulus tuning properties of all neurons"; an example of a maximum entropy model. The performance was very promising, approaching zero error for some stimuli. Another type of transition model, the Leaky Integrate and Fire model Bohte et al. (2000), utilises some of the GLM principles but attempts to model a single neuron in a manner even closer to biological processes - where the current drives a spike, which is emitted when a membrane potential V_i threshold V_{th} is crossed:

$$V_{it} = (1 - \gamma)V_{i,t-1} + I_{it} + \epsilon_{it}$$

... where γ is a hyperparameter that sets how swiftly V_i resets to 0 and ϵ is noise. It is shown to have good accuracy, but is more unwieldy than the GLM.

It is also possible to adapt models discussed earlier by augmenting them with the ability to account for stimuli - in much the same way as the original frameworks can be adapted to account for temporal correlations. Two examples of this approach are the stimulus-dependent Ising model (Schaub and Schultz, 2012), and the stimulus-enhanced restricted/semi-restricted Boltzmann Machine (Kster et al., 2014). In the former, the response probability distribution conditioned on the stimulus is modeled by an Ising model. This requires a lot of training data but seems to work well for a small subset of neurons. Further work is required to verify how applicable it can be to large values of N. The latter involves placing a dependence on the stimulus for the vectors controlling the activity of hidden units, but not involving the stimulus in the interaction between hidden units themselves, helping to limit parameters. A further method is the joint pairwise ME model (Gerwinn et al., 2009), which simply learns the necessary parameters of a maximum entropy model to reproduce the distribution for the joint probability of the stimulus and response, from the data.

Another ME model improved by considering stimuli is presented by Granot-Atedgi et al. (2013). The main contribution of this paper is a model they have called the stimulus-dependent maximum entropy (SDME or S2) model. In particular, it significantly outperforms models that do not consider neuronal pairwise correlation (uncoupled models) when reproducing activity probability distributions during decoding.

$$p(\{x_i\}|\mathbf{s}(\mathbf{t})) = \frac{1}{Z(\mathbf{s}(t))} \exp(-E_{\mathbf{s}}(t)(\{x_i\}))$$

... where $Z(\mathbf{s}(t))$ is a normalisation constant given by:

$$Z(\mathbf{s}(t)) = \sum_{\{x_i\}} \exp\left(\sum_i \alpha_i(\mathbf{s}(t))x_i + \frac{1}{2}\sum_{ij} \beta_{ij}x_i x_j\right)$$

... and $E_{\mathbf{s}(t)}(\{x_i\})$ denotes an energy function:

$$\sum_{i=1}^{N} \alpha_i(g_i(t))x_i + \frac{1}{2}\sum_{i,j=1}^{N} \beta_{ij}x_i x_j$$

Parameters α and β are found numerically using MCMC techniques. The authors highlight that maximum entropy models form each time bin to usually contain only one spike; whilst also containing the important interactions between spikes. This enables viewing of each time bin as a singular observation of neural activity, making the model clearer. Contrasted with a linear decoder, the time bin is chosen by the modeler, usually to be as small as

possible for a higher temporal resolution, no matter the spike count. This is a positive point in terms of interpretability, which was highlighted as a major consideration of this project in chapter 1.

Another important class of algorithm for decoding purposes are classifiers for pattern analysis. One example is by Polyn et al. (2005) which uses a neural network classifier over fMRI data. This imaging technique works with 'voxels' (three-dimensional pixels) representing the whole brain, rather than a singular neuron spike train recorded using a technique like patch clamps. After being trained on brain activity data with known corresponding stimuli, a model was learnt and the network managed classify new activity patterns recorded from the same individual, with statistical significance. Kamitani and Tong (2005) present a research problem of decoding response activity for various rotations of the same stimulus. To achieve this they utilise a linear ensemble decoder, which assigns a separate decoder for each orientation. The final prediction is made by selecting the most active detector in the ensemble. They achieved results of 96.1% accuracy. Brain activity in the context of lie detection is explored by Davatzikos et al. (2005), who use a non-linear support vector machine (SVM) with a Gaussian kernel. After being trained on existing data and learning a model, remarkably this technique approached 100% separability between truth and lies when operating on unseen brain activity. Finally, Norman et al. (2006) gives an overview of the multi-pattern voxel analysis (MVPA) approach, which is a framework within which many different types of classifier can be used. Instead of spatially averaging across voxels in fMRI data, this model can work on entire data sets to increase the signal-to-noise ratio, and in their words 'mind-read' through pattern analysis. However, many of these models leave a lot to be desired in terms of interpretability. For example, SVMs utilise the kernel trick to facilitate dimensionality reduction in data, making it hard to relate the model's output back to the original data. Neural networks are comprised of hundreds, if not thousands of nodes, all with their own parameters learned from data. Furthermore, they are not probabilistic and cannot deal well with uncertainty. Additionally, whether these models can also work with the type of spike train data that will be studied in this project as well as fMRI data is a point for further investigation.

Another publication utilising linear pattern analysis algorithms is by Esfahany et al. (2017). The dataset used in the study is the Allen Institute data and the study decodes one of six visual stimulus categories with distinct structures. An important conclusion drawn in this study is that the accuracy of the decoders was different for each visual area of the brain, which the authors state implies differential information representation in these areas. The results also point towards the neurons in superficial brain layers being more informative about stimulus categories than neurons in other layers. A limitation of the study is that there is no interpretive model analysis to discover what properties are important for the task of decoding; they are simply learnt, then used and tested. Therefore this study cannot provide good insight on how information is represented in the brain, only on the model itself. There is also no further comparison of models beyond the simple linear decoder. Finally no comment is made on the applicability of the system to larger/different datasets.

4.3 Conclusion

This review has highlighted multiple techniques from a range of approaches viable for investigating the neural code, each of which has its own advantages and disadvantages, and apply to different types of data. The most significant body of work is being done under the ME framework, which has many advantages including its probabilistic nature and its

flexibility in terms of possible constraints. However under certain conditions the parameters can still grow quickly, casting doubt on tractability and interpretability. A number of more general, common ML techniques such as neural networks and other pattern recognition approaches are also being applied to neural data, but it seems their disadvantages are too great to consider them seriously in a study focusing on interpretability, which almost all of them face issues with. Conditioning on stimuli or temporal properties of the data can be done with varying difficulty, providing marginal gains in performance, but most of these techniques seem to be variations on existing models and frameworks rather than being developed in their own right.

Clearly there are too many models to implement and investigate in this one study, so one of the many decisions to be made prior to implementation will be choosing a subset to take forward for further exploration.

The survey has also provided insight into what a framework for assessing the models might look like. One's intuition for the criteria for deciding the 'best' model for encoding/decoding neural data might be the model that provides the highest accuracy. However, it is clear from the lack of comment on model interpretability in the surveyed literature that this is an area which needs significant attention. It should be asked where the main value in these models actually lies: is it in performance alone; or does the knowledge we can extract about the underlying system being modeled matter more? Perhaps the main value in some of these models lies in the information that can be extracted from it through human interpretation and understanding.

Part II

Execution

Chapter 5

Methodology

5.1 Technology Stack

There were a number of decisions to be made regarding the technology chosen to facilitate the processing and analysis of this large dataset. The programming language underpinning all of this was Python, for three main reasons. Firstly, its large ecosystem, especially for data science. This includes a multitude of packages with useful code available for reuse (outlined below), as well as a large support community with many questions asked and answered online. Secondly, data wrangling is very easy in Python - reshaping, slicing and file I/O can usually all be done in a single line of code. Thirdly, whilst vanilla Python has poor performance overall when compared with a low-level language such as C, it can be enhanced significantly with a range of techniques. Taking a simple operation as a rough example, computing standard deviation on an array of values can be sped up by 30x using NumPy, and as much as 80x using Cython.

The packages used are outlined in table 5.1. For the most part they are common packages used in many data science applications. The key package to note is the `allensdk`, which provides an interface with the servers that host all the data from the Allen Institute experiments as described in section 1.5. Functions are provided to query metadata (see listing 5.1), helpful to enable identification of useful experiments prior to downloading the hefty datasets. Data in the form of NWB files are also downloaded using the SDK, orchestrated by the 'Brain Observatory Cache' - a JSON file which keeps track of the data already downloaded locally, to prevent re-downloading data if it already exists. The NWB files are wrapped in Python objects that provide functions to isolate specific parts of the data; for example dF/F traces or motion-correction data for specific experiments. Finally,

Packages	Purpose	Reference
`allensdk`	Querying neural data	(Allen Brain Institute, 2018)
`numpy`	Mathematical operations, fast arrays/matrices	(Oliphant, 2015)
`pandas`	Flexible, versatile `DataFrame` data structure	(McKinney, 2010)
`matplotlib`	Plots & figures	(Hunter, 2007)
`jupyter`	interactive scripting	(Kluyver et al., 2016)
`scipy`	Statistical operations & Matlab integration	(Jones et al., 2001)
`c2s`	Spike inference	(Theis et al., 2016)
`sklearn`	Data splitting, ML algorithms, accuracy scoring	(Pedregosa et al., 2011)

Table 5.1: The packages used during data analysis.

a set of precomputed cell metrics are provided, highlighting interesting cells found during the experiments - for example neurons highly sensitive to grating orientation.

```
{'age_days': 101,
 'cre_line': 'Cux2-CreERT2/wt',
 'device': 'Nikon A1R-MP multiphoton microscope',
 'device_name': 'CAM2P.2',
 'excitation_lambda': '910 nanometers',
 'experiment_container_id': 511510779,
 'fov': '400x400 microns (512 x 512 pixels)',
 'genotype': 'Cux2-CreERT2/wt;Camk2a-tTA/wt;Ai93(TITL-GCaMP6f)/Ai93(TITL-
    GCaMP6f)',
 'imaging_depth_um': 275,
 'indicator': 'GCaMP6f',
 'ophys_experiment_id': 503019786,
 'pipeline_version': '2.0',
 'session_start_time': datetime.datetime(2016, 2, 21, 17, 41, 47),
 'session_type': 'three_session_B',
 'sex': 'male',
 'specimen_name': 'Cux2-CreERT2;Camk2a-tTA;Ai93-225036',
 'targeted_structure': 'VISp'}
```

Listing 5.1: An example of experiment container metadata exposed by the `allensdk`

5.2 Model Shortlist

In light of the literature survey in chapter 4 and after discussion with the project's supervisor, a shortlist of models was constructed from the longlist generated in the literature survey. The shortlist included:

- Independent model of neurons
- Linear decoders (Support Vector Machine / Multinomial Linear Regression)
- Population tracking model
- Restricted Boltzmann Machine

These particular models were chosen because they cover a range of approaches, including linear models, models utilising the maximum entropy framework, and probabilistic approaches. They are also well distributed across various points along a triadic reciprocity balancing accuracy, computational cost, and interpretability (see figure 5.1).

5.3 Code Familiarisation

Time was allocated in the project plan prior to data analysis for code familiarisation. A range of packages are being utilised, all with their own methods and objects. Additionally, some of the code only exists in MatLab and will need to be translated to Python or, if this process is deemed inefficient, used as-is. The importance of early familiarisation with new code and environments should not be understated, as knowledge of certain techniques can significantly speed up development further down the line.

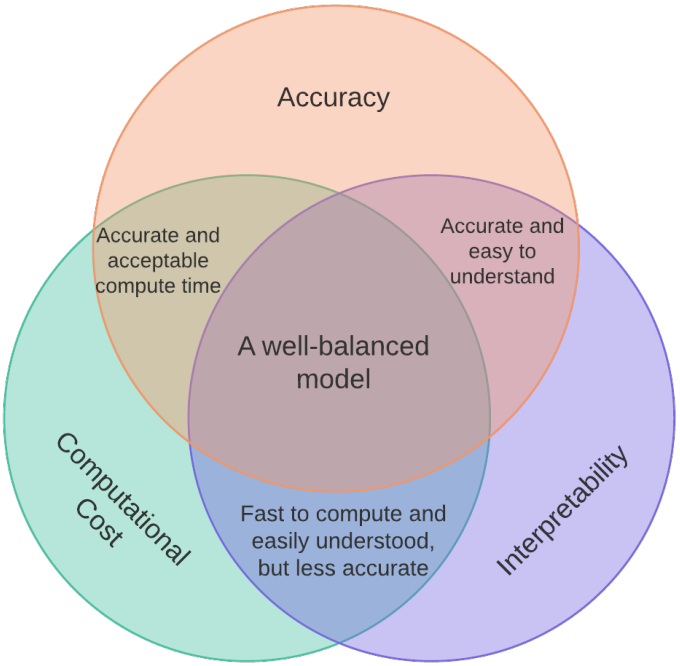


Figure 5.1: The models chosen for exploration distribute well across a scale balancing accuracy, performance, and ease of understanding. From the literature review, most current models seem to fall in the area between accuracy and computational cost.

To engage with data analysis in Python, a range of tutorials were practised with. These included building a simple model to classify types of glass, given the quantity of various elements present in the glass; using a 'bag of words' model to classify complaints lodged with the US Consumer Financial Protection Bureau in text form; and using multinomial linear regression techniques to try to predict whether a person will have an affair based on features extracted from a large dataset constructed from responses to a questionnaire about marriage satisfaction.

Finally, the default Matlab installation includes a range of example data and scripts which were used to learn Matlab specific operations. These were explored in-depth across several domains in order to become comfortable with the Matlab IDE environment and syntax.

5.4 Data Exploration

In order to solve problems with data, it is important to know what data to actually use to find the solution we are looking for. Blindly searching through massive datasets is inefficient if techniques exist to easily identify that the solution exists in a small subset. Data exploration can also reveal important properties of the data under scrutiny, such as the size and dimensionality (Idreos et al., 2015).

To explore the Allen dataset, a Jupyter notebook was created with the eventual goal of producing some basic visualisations over the data. The first step to achieving this goal was exploring what metadata was available, in order to identify useful experiments. The Brain Observatory Cache (`boc`) was initialised which creates an empty `manifest.json` file which will be used to track data downloaded. If a function is called that requires data to

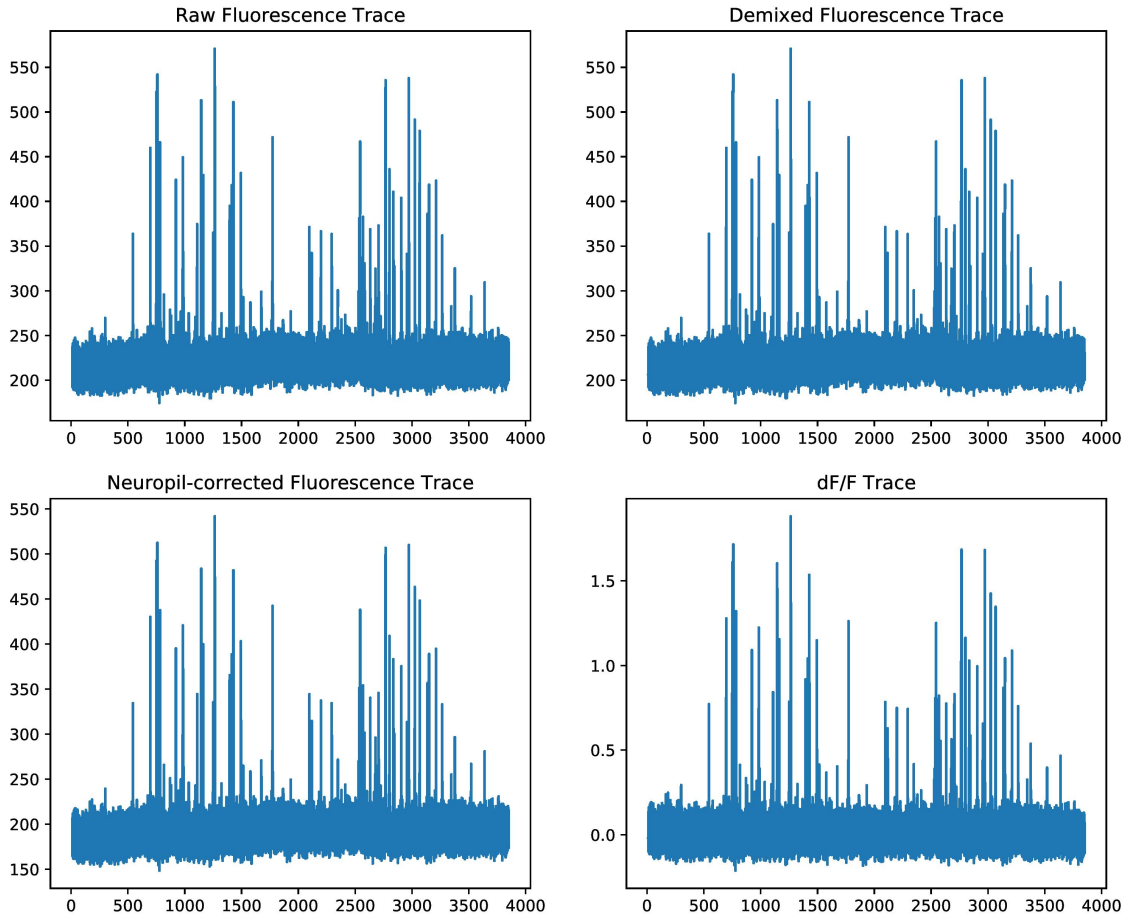


Figure 5.2: Some of the trace data available. Clockwise beginning top-left: *1.* recorded trace. *2.* trace corrected for spatially overlapping cells. *3.* trace corrected for non-cell-body entities. *4.* change in fluorescence intensity dF/F plot.

be downloaded from Allen Institute servers, the `boc` will first be queried to check if a local record exists, and if so, it is this copy that will be used. A simple example of the metadata that can be exposed by the `boc` is the data returned by the `boc.get_all_stimuli()` function. This returns a list of strings describing the types of stimuli used across all experiments in the dataset.

Drilling further down, individual experiment metadata can be retrieved by acquiring the containers for those experiments with the `boc.get_experiment_containers(args)` and passing in a container ID, a stimulus, or another property to filter the experiments by as the function argument (see listing 5.1). In this instance the specific experiment used was not important, so an arbitrary ID was chosen. After the NWB file was downloaded with `data = boc.get_ophys_experiment_data(id)`, specific parts of the data can be isolated - including raw, demixed and dF/F traces.

Most functions based on the data return two vectors: one containing the data itself (dF/F values etc.) and the other containing the corresponding timestamps. This makes it easy to plot returned data, as the timestamp is simply assigned to the x-axis and the values to the y-axis (see figure 5.2). Plotting was done with the `matplotlib`. This package provides many different types of plots to visually represent data and provides fine-grained control over things such as colours, titles and plot sizes.

A further visualisation was a region of interest (ROI) mask. This displays the location

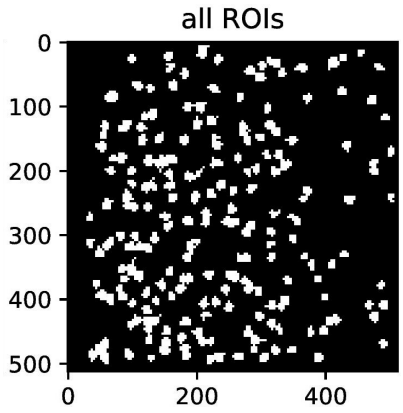


Figure 5.3: Region of Interest (ROI) masks for neuron activity - the location within the field of view of the imaging device of the active neurons.

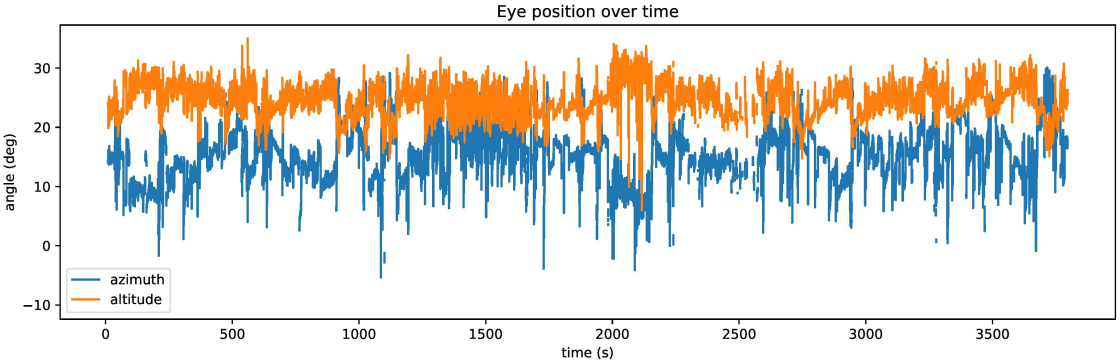


Figure 5.4: Mouse eye movement during the experiment.

of the active neuron within the viewing field (512x512 pixels as specified in the Allen Institute experiment whitepaper). All cells can be plotted to create an overall ROI mask - see figure 5.3. Finally, ancillary information such as motion-correction and eye-position during the experiment can also be plotted (figure 5.4).

Overall this represents quite a large amount of learning and information explored. This was useful because it narrowed the amount of data we needed to look at during analysis and increased efficiency. An example of how the exploratory process expedited data analysis is through use of filters for the functions that return data, which was a feature of the SDK not known prior to this exercise. It is possible to filter on container, experiment, or even cell ID. An example of how the plots aided analysis is shown in the corrected fluorescence trace plot vs. the dF/F plot. Whilst their shapes look similar, the plot exposes the difference in scale, highlighting why dF/F is more useful than fluorescent magnitude alone.

After an acclimatisation period learning the data makeup and how the code is put together, the true data analysis could begin. Most of the models required the same data pipeline, which at a high level consisted of data acquisition, data pre-processing, model construction, and results analysis.

5.5 Data Acquisition

The entire dataset was downloaded to local storage. This provided flexibility, as even though it would likely be filtered later, it was not totally clear exactly what data would be used to extract features, so having all data available was useful. All containers were downloaded but it became apparent the servers were rate-limited. The download script was therefore adapted to split the downloads across multiple threads and requests to saturate the download rate. This was achieved with a simple function `boc.get_ophys_experiment_data()`, which attempts to load the data into memory by querying the `boc` to check if the data exists locally - downloading it if not. The process of executing this function over each container ID, even though nothing was actually done to the data after the function had finished executing, meant that all data now existed locally. This totaled approximately 170GB.

5.6 Processing pipeline

After acquisition, the main script began. Jupyter Notebooks were used due to their modular nature. This structure suited the project excellently as lengthy scripts processing dozens of GB of data were often run, and should there be any runtime errors or modifications required to the flow of the script, this could be easily done without having to start completely from scratch.

5.6.1 Pre-processing

The first step towards data analysis was to construct a list of viable experiments from the master list of containers downloaded in the previous steps. This itself comprised three distinct stages run for each container in turn. Firstly, a list of cells that appeared in all three experiments within the main containers (sessions A, B, and C/C2) were filtered for, so cells that were, for whatever reason, only imaged in one session, were excluded. This was important to eliminate any bias towards a particular session which may have included unmeasured variables not controlled for, despite being essentially the same. Next, a check was run to ensure this filtered list included at least ten cells - any fewer was deemed too small a sample size. The final check was to ensure that within the cell data itself there were no null or missing values. Whilst the occasional missing value may not have presented too much of an issue in terms of interfering with model accuracy, they are hard to check for and ignore at each step of data processing and it is easier to simply discard containers with these characteristics.

Overall this resulted in excluding ten containers with too few cells, and one container with missing values, for a total of 170 containers remaining. It should also be noted that in the Allen dataset itself, one additional check had already taken place which resulted in the removal of some experiment containers prior to even downloading the data locally. Steinmetz et al. (2017) uncovered abnormal neuronal activity under some conditions in some mice with the same genetic modifications as those used in the Allen Institute study. Therefore, the Allen Institute identified experiments exhibiting this abnormal activity and has since excluded them from the dataset.

5.6.2 Feature Extraction

Following pre-processing, most models required some filtering of data to direct the model in exactly what data the model should be looking at to uncover patterns - known as 'features'. The process to identify features is known as feature selection when they are simply selected from data, and feature extraction if some form of data manipulation has to take place to isolate them. In this case, extraction was mostly required. Each experiment consisted of three hours of traces, split into stimulus epochs.

5.6.2.1 Linear Decoders

Broadly, extraction for the linear decoders involved generating a labeled mean fluorescence magnitude across a set of pre-defined time bins for each cell in each experiment. Fluorescence intensity (dF/F) traces were isolated from the data. The goal was to split these into samples of mean dF/F over ten second intervals. To achieve this, start/finish times were calculated from the returned stimulus epochs and matched with the corresponding start/finish values in the timestamps. These pairs were then labeled with a corresponding stimulus value from a list of all stimuli used in the experiments.

After time-binning, the dF/F values underwent a process of z-scoring using the `scipy.stats.zscore()` function. This was done to smooth any variation in the distribution of the fluorescence values across time and between sessions, as the experiments were conducted over a lengthy period of time. These were then combined into an overall feature array, which constituted the feature set of considerable size - 12,123,614 data points.

5.6.2.2 Population Tracking

For this model, the dF/F traces would also be required, but since they were being fed into the model in a very specific manner rather than going on for further processing, the structure was the key consideration here. The population tracking model requires input of an $N \times T$ array where N is the number of neurons and T is the number of timestamps. Therefore for each container, and in turn for each of the enclosed experiments, the cells were arranged in a multidimensional `numpy` array, all of which were in turn arranged into a master list for portability, which could be indexed to query individual experiments. Therefore for each experiment, many less data points existed compared to the linear decoders, which is an advantage of this approach.

5.6.3 Data Preparation

5.6.3.1 Linear Decoders

Now the feature set had been constructed, it was necessary to split this data into training and test sets. The models were constructed over the larger training set, covering 80% of the available data. The remaining 20% test data set was used to provide an unbiased evaluation of the model. The purpose of splitting the data in this way is well documented (Bradley, 1997), (Batista et al., 2004), but briefly, this process exists to avoid overfitting. Overfitting can occur when a model too closely models the underlying system, therefore likely capturing noise in the training data. These noisy values do not accurately describe the system in general, but nonetheless the overfit model performs very well over the data

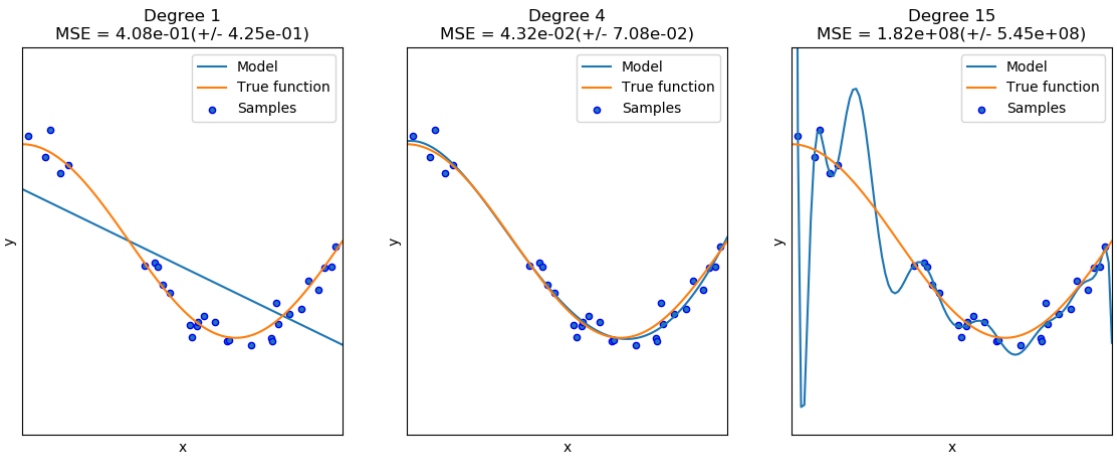


Figure 5.5: *Left:* underfitting - learned function (blue) does not have enough degrees of freedom to capture the underlying system (orange). *Centre:* good fit. *Right:* overfitting - learned function tries to match data too closely resulting in poor generalising capability. *Source: (Pedregosa et al., 2011)*

it was trained on - but generalises poorly to unseen data (see fig 5.5). The unseen test set helps verify this is not the case. If overfitting has occurred, hyperparameters can be adjusted, or training can be stopped earlier.

In most cases, the imbalance of the data precluded direct splitting of the feature set. For example, the number of data points available for some classes of stimuli was significantly lower than for others. This necessitated balancing of the datasets to avoid biasing the model towards one set, which was done at this stage. The smallest dataset was found (spontaneous stimuli), and all other training datasets were scaled in line with this size (so eventually, all datasets were 80% of the size of the smallest dataset).

5.6.3.2 Population Tracking

Similarly, now the data has been isolated and extracted, the final stage was to prepare it for the constraints of the population tracking model itself. This particular model is constrained to binary data in the form of spike trains - an $N \times T$ array of time separations where the neurons were either active (1) or silent (0). This spike train must be extracted from the fluorescence trace provided by the `allensdk` in a process known as spike inference. As explained previously in section 2.1.1, recorded fluorescence does not map directly to spike activity as the fluorescence decays at an exponential rate, in a much slower fashion than the almost immediately on/off neurons. Therefore a model must be used to infer the spike activity if the model requires that data directly.

A number of packages were explored, including `FastLZeroSpikeInference`, `PyFNND` (Python Fast Non-Negative Deconvolution), and `c2s`. All of these presented a technical issue as work had primarily been undertaken on a Windows PC up to this point, but these packages only supported Unix systems. This raised another issue - how to port the data processed in Python up til now across systems, which mostly resided in Jupyter Notebook scripts rather than permanently on disk. This required use of the `numpy.save()` function, which binarises NumPy arrays and saves them to disk using Python's `pickle` package, ready to be loaded later with `numpy.load()`. This produced around 20GB of compressed data to be transferred, which was then processed by the chosen package `c2s` as it seemed

to have the best integration with Python, whereas the other packages were well-suited to Matlab with Python added later, sometimes in beta or even alpha state.

`c2s` took care of 'pre-processing' (normalising) the spike train, followed by 'predicting' - using a model previously trained on 110,000 spike trains, the details of which can be found in Theis et al. (2016). The predicted spike train was binarised and exported to a Matlab file, retaining the structure architected in section 5.6.2 using `scipy.stats.savemat()`. This was necessary because the implementation of the population tracking model was in Matlab.

5.6.4 Optimise Hyperparameters

After splitting the data to enable model construction, any hyperparameters necessary for the model must be chosen. Hyperparameters refer to parameters of the model's learning process ('meta' parameters); the parameters of the model itself are simply known as 'parameters' and calculated during training. In many cases this primarily consisted of a regularisation constant.

Regularisation was applied to the linear decoders as another technique to reduce the tendency to overfit training data. As models begin to overfit, their complexity also increases significantly, and as a symptom of this the parameter (weight etc.) values increase suddenly. Regularisation simply applies a penalty to large parameter values, thus penalising overfitting. It reduces variance without substantial increase in bias. There are various types of regularisation available, but in this instance 'L2' or 'ride regression' was used, which takes the following form:

$$C = L + \frac{\lambda}{2m} * \sum ||w^2||$$

...where C denotes the cost function as a whole, and L is the original loss function component. From the above equation it is clear that L2 regularisation forces the weights w to decay towards 0 by adding a squared magnitude coefficient as a penalty, with λ as the hyperparameter that can be adjusted to ensure a good fit.

One of the techniques used to optimise the hyperparameters was cross-validation. This process, whilst computationally expensive, is a simple way of selecting a model variant for a problem with minimal error rate, minimising estimation bias (Weiss and Kulikowski, 1991), (Breiman, 2017). Data is split into a number of groups - the more groups, the higher the computational cost but the more effective the validation process. In this case, five-fold cross-validation was chosen. For each iteration, one of the five groups is reserved for test whilst the remaining data becomes the training data. This is repeated cycling through the groups. This has the effect of using most of the data for training, and most of the data for validation - reducing the chance of underfitting and reducing variance. This was done for a range of constants $C = \{10^-2, 10^-1, 10^0, 10, 10^2, 10^3, \infty\}$, and the model with highest accuracy was identified and the corresponding C value was selected to use as the regularisation constant.

This method was chosen because the data set has been reduced through balancing and it was preferred to not sacrifice any further data by splitting permanently. Furthermore this gives better accuracy as the process is repeated multiple times.

5.6.5 Build Models

The final stage after all the above was to actually build the models, to enable predictions to to be made which would facilitate an assessment of their accuracies. For the linear decoders this meant fitting the data to the specified model. For the population tracking model, this involved fitting the parameters to the model, a process which was briefly discussed in section 4.1. This results in an output of an estimate for probability distributions $p(K)$, $p(x_i|k)$, and $p(x_i)$ (mean firing rate).

It should be noted that for the linear decoders, subsampling from the neuronal population took place without replacement for various population sizes $n = \{2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7\}$ to investigate how the models' accuracies changed as a function of population size N. The procedure was repeated 10 times to improve accuracy.

5.7 Conclusion

Quite a sophisticated data science pipeline was applied in the section of the project. Despite significant effort invested into research into models, in reality this is mostly dealt with in the theoretical side and when it comes to implementing them, existing solutions work well or implementations are fairly trivial due to the detailed formulation in theory. The vast majority of effort in the execution of the stages above, which is evident from the division of focus on each of them, is the data wrangling. Acquiring, transforming, manipulating, and processing this data into a suitable form to use as inputs for the models can be a nontrivial task.

This is amplified when catalysed with technical complexities. For example, many of the code used above was not platform-independent, requiring a complete change of operating system at one point, which introduced friction into the execution process (see section 8.4). Another of these technical challenges was the fact that whilst most of the work was undertaken in Python for the reasons mentioned in section , the population tracking model implementation existed in Matlab. A fairly crude Python port exists, but upon closer inspection it was decided this would not be suitable as a number of features seemed incomplete, and the port adds another layer of complexity which should be avoided. Creating an original port to Python was also considered, but due to the timeframe was not pursued and is instead left as future work (see section 8.3).

Part III

Results and Analysis

Chapter 6

Results

6.1 Accuracy Calculation

The final step in the processing pipeline was to output the results. For the linear decoders, accuracies were calculated and a range of graphs were produced for various different combinations of brain regions, imaging depths, and stimulus classes. The population tracking learned a model from the data and reproduced various distributions describing the population data, which were evaluated with the following methods.

6.1.1 Linear Decoders

For the linear decoders, accuracy calculation was a fairly straightforward case of measuring accuracies using the `sklearn.metrics` package. For example, the `accuracy_score` function uses the learned model to make some predictions from the test set, and measures the classification accuracy against the actual data. The discrepancy is the error.

Since data for the linear decoders was also subsampled to investigate accuracy as a function of population size, an extrapolation function was required because whilst the subsampling occurred up to the highest number of imaged neurons available for each mouse, this was not the actual number of available neurons at all times for all experiments. Some experiments contained fewer neurons. The Allen Institute does not make it clear why this is the case, but it could be due to different setups in imaging sessions, some neuron imaging was not effective for some experiments, or they were removed due to errors such as the aberrant gene expression noted in 5.6.1.

Therefore, the calculated accuracies of the models constructed up to the actual available number of neurons needed to be extrapolated up to 128 in some occasions for some models. Esfahany et al. (2017) selected the generalised logistic function:

$$\text{accuracy} = \frac{1-c}{(1+\exp -an)^b} + c$$

... where $a \geq 0$, $c \geq 0$ and $b \in [0, 1]$.

6.1.2 Population Tracking Model

Assessment of the accuracy of the population tracking model is somewhat less straightforward. The output of the model comes in two primary forms, corresponding with its two

primary components: $p(K)$ (probability distribution for the population rate) and $p(x_i|K)$ (probability of neuron i activity, given the population rate). It is less easy to compare this type of model with data as in the linear decoder accuracy assessment because the discrepancy will be somewhat different. Therefore we can turn to the concept of entropy to judge accuracy, and compare the reproduced distributions with that of another model.

To compare two probability distributions $p(x)$ and $q(x)$, the population tracking model facilitates the use of the Kullback-Leibler (KS) divergence or the Jensen-Shannon divergence (JS). Both of these metrics measure the similarity between two probability distributions; the latter extending the former to apply to additional situations, but also bringing some limitations. For a discrete random variable X, KS divergence $D(p||q)$ takes the form:

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

It heavily penalises regions where $p(x)$ and $q(x)$ do not overlap, making it extremely useful when trying to find estimates for intractable distributions, and it must be ensured that samples unable to be drawn from the true distribution are also very unlikely to be from the approximation. The JS divergence uses this concept as a building block but has another defining property:

$$JSD(p||q) = \frac{1}{2}D(p||m) + \frac{1}{2}D(q||m)$$

... where $m = \frac{1}{2}(p + q)$. Therefore JS weights two distributions evenly, resulting in symmetry, excellent at purely measuring distance between distributions.

One good comparison baseline to use is the model of independent neurons, as this is a simple model which makes few assumptions and is quick to compute. It is easy to interpret and assess - the fact it is unable to capture much structure in neural population data due to its inherent assumption that correlations are not present does not particularly matter for this purpose. Another useful way of applying these techniques of measuring the difference between distributions is not to assess the model itself, but to compare neural activity during two different sets of conditions, to measure the change in conditions' effect on the activity.

6.2 Performance Summary

In figure 6.1 the accuracies for the SVM are plotted as a function of population size, separated by brain region. This plot investigates whether the SVM performance changes depending on the area of the visual cortex being modeled, to explore adaptability and applicability.

Figure 6.2 portrays the performance for the SVM, again plotted as a function of population size, this time divided by stimulus direction. The data in question is from experiment session A only, as that is the only session where the oriented gratings were shown. This plot investigates whether the decoder performs differently across the various groups of neuron that might be directionally tuned. Figure 6.3 shows a similar SVM again built to decode grating direction, but this time the dataset has been shuffled to remove temporal aspects and attempt to introduce independence.

Finally for the linear decoder, figure 6.4 shows the accuracies for the SVM plotted as a function of population size, separated by imaging depth. This investigates whether the performance changes depending on the level at which the images neurons lie in the brain.

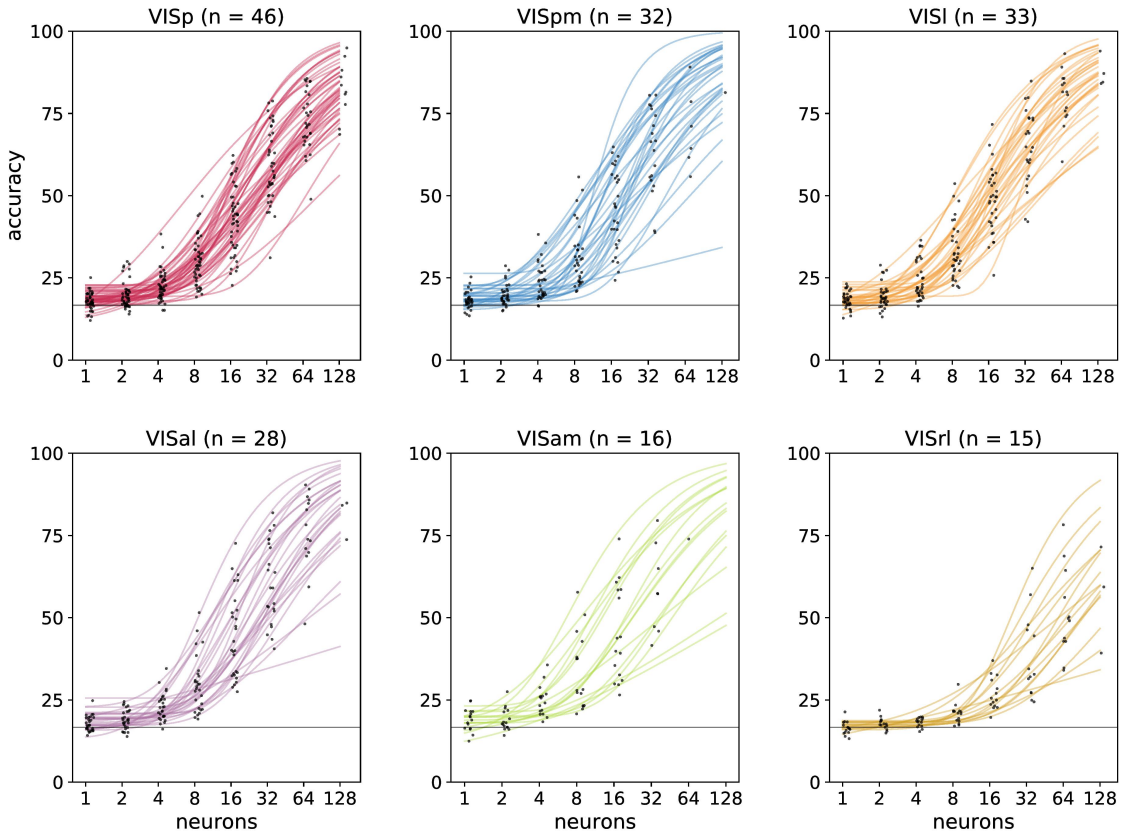


Figure 6.1: SVM accuracy when decoding stimulus data, as a function of population size, separated by data from the six V1 regions

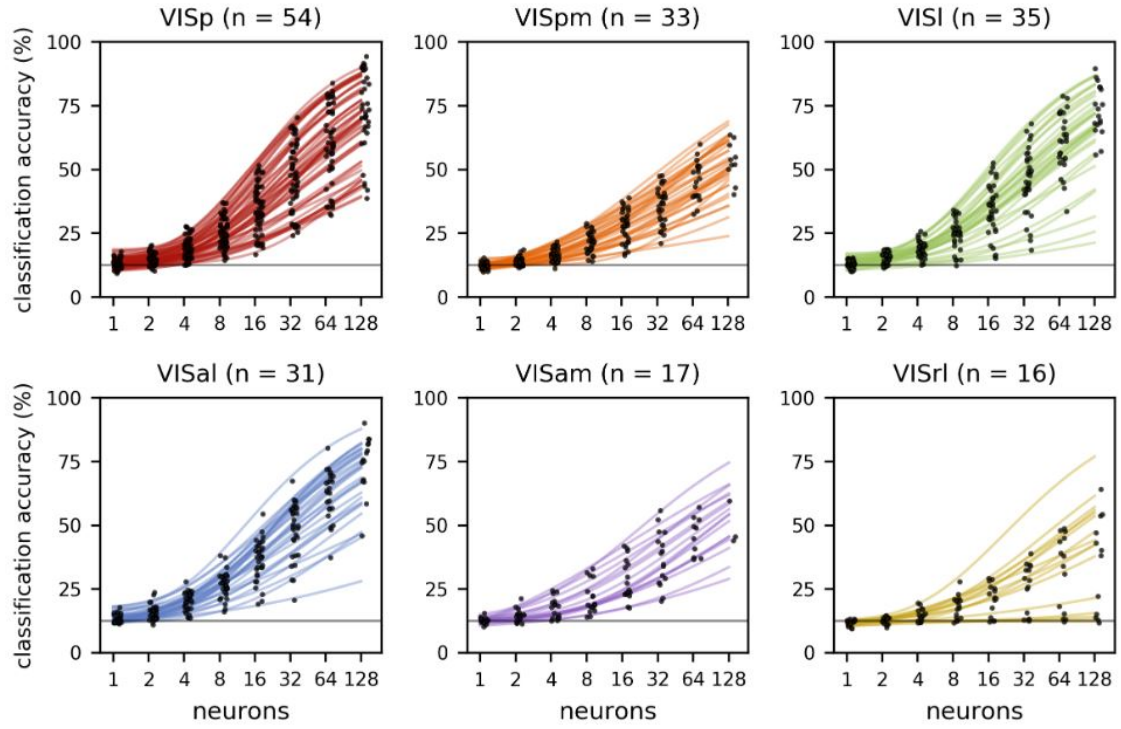


Figure 6.2: SVM accuracy when decoding grating direction data, as a function of population size, separated by data from the six V1 regions

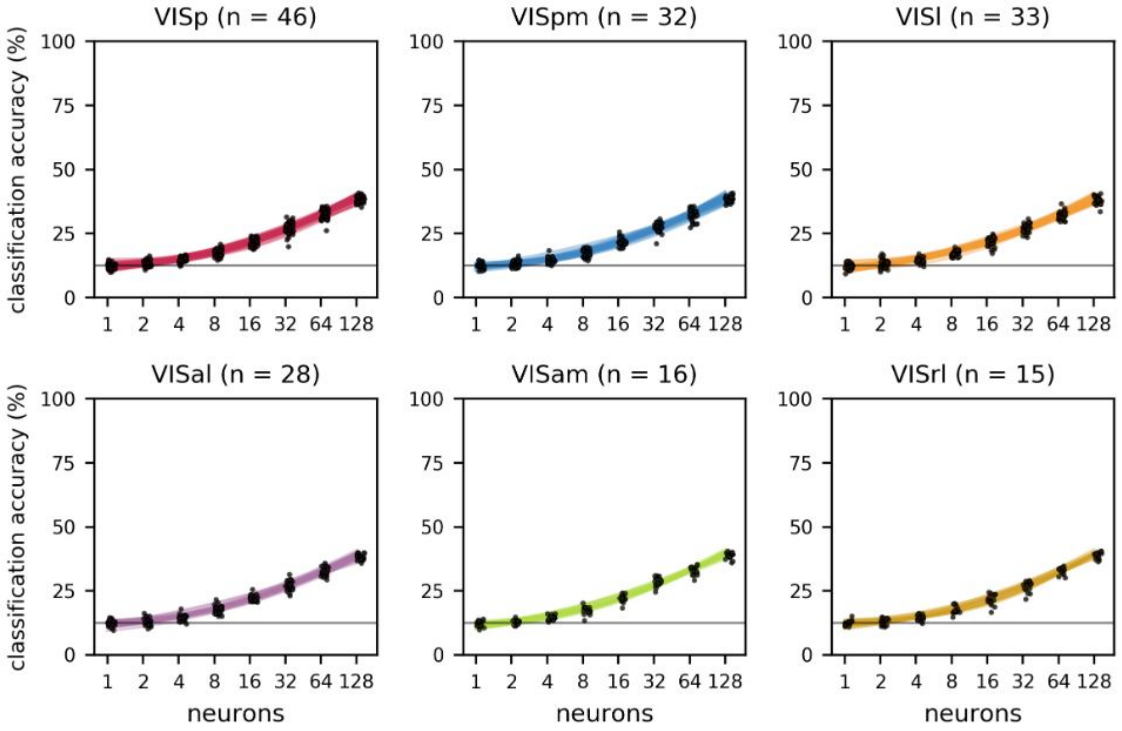


Figure 6.3: SVM accuracy when decoding shuffled grating direction data, as a function of population size, separated by data from the six V1 regions

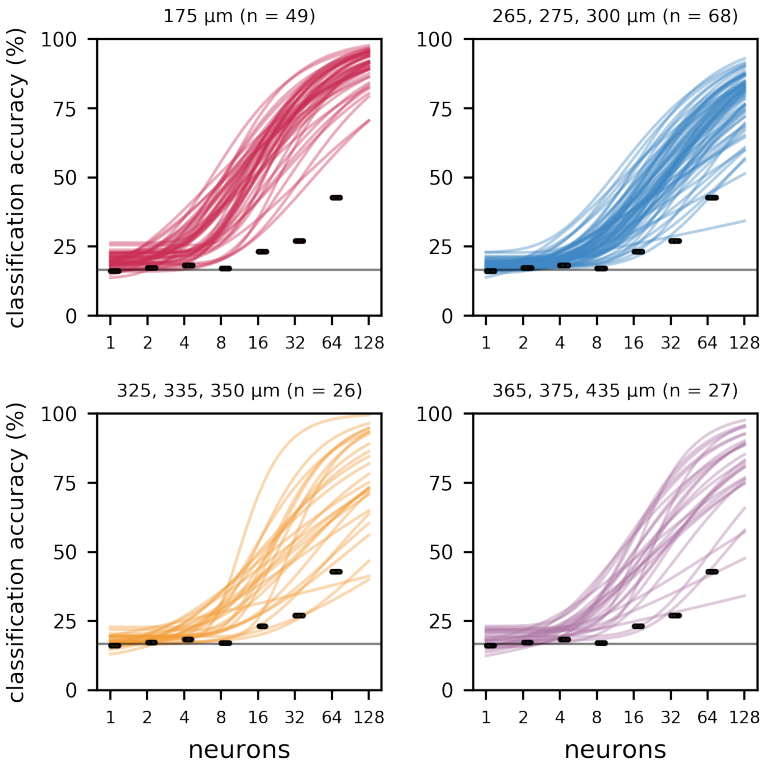


Figure 6.4: SVM accuracy when decoding stimulus data, as a function of population size, separated by data from four groups of imaging depths

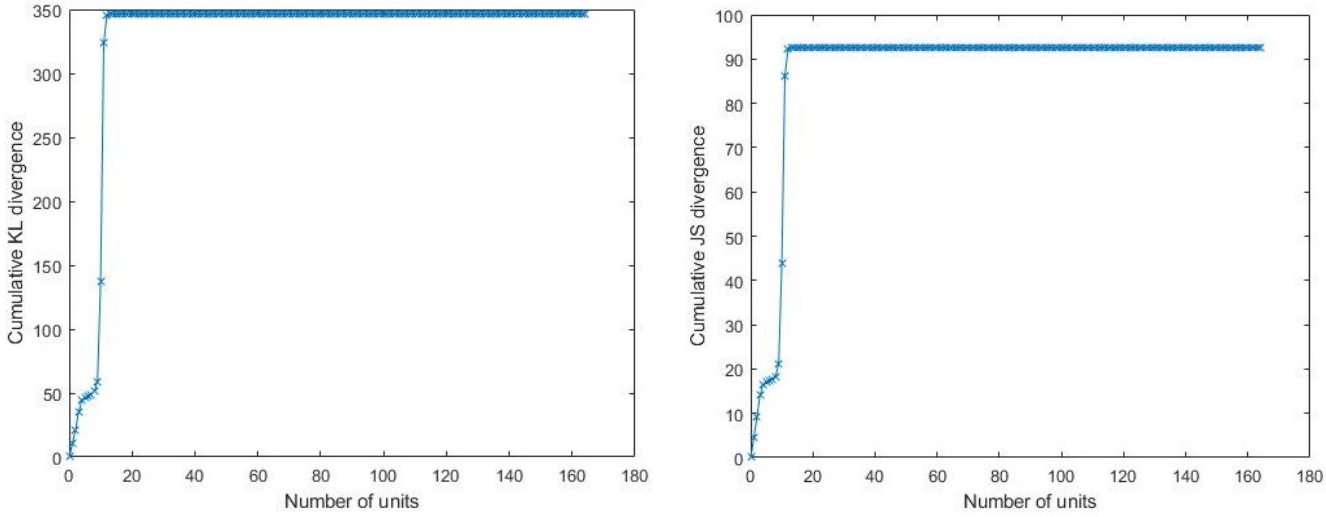


Figure 6.5: For population tracking vs. model of independent neurons - *Left:* KL divergence *Right:* JS divergence

The KL and JS divergence for the population tracking model versus the model of independent neurons can be viewed in figure 6.5. An example of the concept discussed earlier, comparing neural activity under two sets of conditions, can be seen in figure 6.6. In this instance, the plots compare neural activity from distributions built over data from two different session types (A & C) containing different stimuli, to examine the effect this change in stimuli has on the response.

6.3 Conclusion

After the lengthy data pipeline in chapter 5, the outputs of the models are condensed into succinct and efficient plots. All these plots are based on large matrices of accuracy scores and probability distributions, which would be useful in an analytical sense if being used to be processed further. In this instance however, a series of summary visualisations are more useful to enable interpretation of the results. These will be discussed in the following chapter.

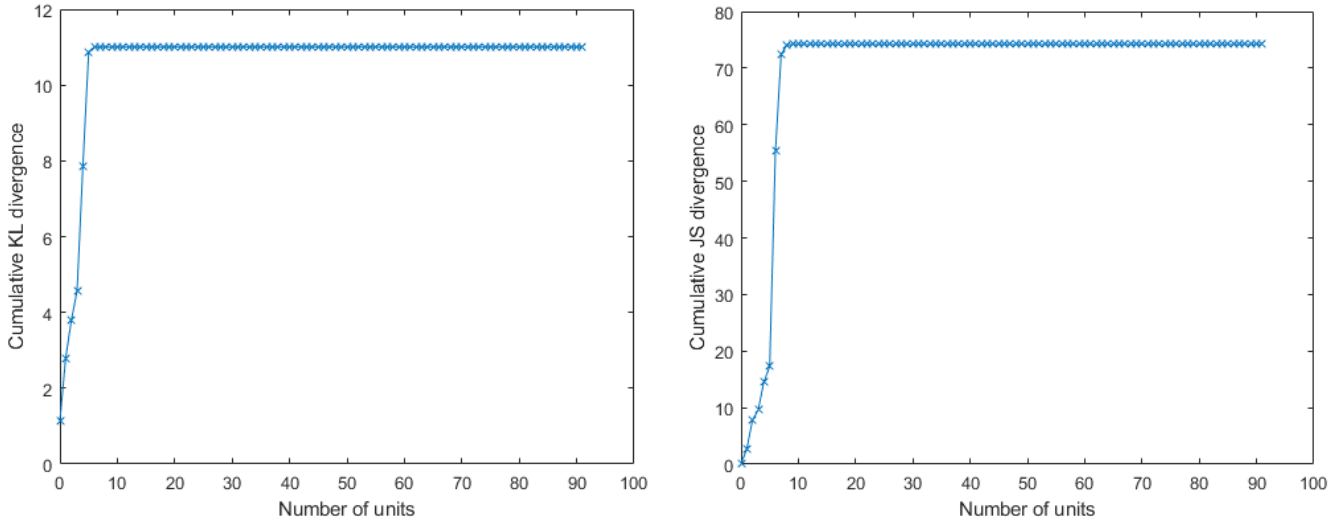


Figure 6.6: Population tracking models learned over session A vs. session C - *Left:* KL divergence *Right:* JS divergence

Chapter 7

Discussion

In the previous two chapters, the data was processed, the models were constructed, and the outputs in the form of accuracies and reproduced distributions were produced. This entire process exists to facilitate a comparison of the systems to judge how well they have modeled the neural code, and to comment on that code. The comparison follows in this chapter.

7.1 Accuracy Evaluation

The linear decoder accuracy when decoding input data in general was very good. It is clear that across all six brain regions, for all types of data, the accuracy increases as the population size increases, which is as expected. As the population size increases, the available data increases in richness, as relationships between neurons can be captured in size. The amount of information contained within this data also increases, as it moves away from an independent model up towards 2^N (although obviously still a long way off). The range of lines on the graphs is due to repetitions of experiments, within which the performance does vary quite a bit. This is somewhat concerning, as if the model is capturing a generalisable, broader model of the neural code, the particular experiment should not matter. At the very most, separation would not happen in so many variants; instead we would perhaps expect to see three separations, one for each session in which different stimuli were shown. Black points on the plot are individual experiment results.

Moving through each of them in turn, figure 6.1 shows good performance across all regions. There are some anomalies, such as a very low performer for VISpm visible in the bottom right of the blue plot. This doesn't seem to have any black points after $n = 8$, so this could be because the experiment contained very few neurons and therefore the low performance is due to the accuracy extrapolation curve rather than the decoder performance. VISrl performs noticeable more poorly than the other regions, with several experiments falling around 50% accuracy even for large populations such as $n = 64$ or even $n = 128$, which is in keeping with current papers. It is not yet clear why this region performs worse.

For figure 6.2, again VISrl is the outlier but this time VISpm also seems to perform slightly worse than the other four regions. Many experiments again hover around 50% accuracy. This difference in performance for a single brain region across two types of stimuli is particularly interesting because this may indicate that this region responds mostly based on stimuli type, and is not particularly direction-selective, showing differences amongst

brain regions. VISp performs particularly well with some experiments approaching 95% or higher.

Figure 6.3 is an outlier amongst these plots as it does not reproduce the findings of other current research. For shuffled datasets which attempt to remove a temporal element and introduce independence, performance is usually maintained with linear decoders, suggesting temporal information is not as important - which is to be expected as they mostly deal with classification. However clearly these plots show the accuracies as being significantly worse - still increasing with population size so capturing some relationships, but not nearly at the scale of the others. It remains unclear why this was the case for this project. The issue may lie in data pre-processing or the shuffling algorithm.

Figure 6.4 is a satisfying plot as it lines up quite perfectly with one's intuition regarding imaging depth. The performance is surprisingly good across all depths, approaching 100% for the shallowest set. Looking at the plots, it is clear the accuracies decrease for each plot, as the imaging depths get deeper. This is probably to be expected as one would expect it is more difficult to accurately image neurons deeper in the visual cortex, due to neuropil interference and general distance. The consistency across experiments is noticeably worse for the yellow plot, suggesting the fault lies with the experiment rather than the decoder. Finally, the purple plot (deepest imaging) clearly has the worst accuracy.

The population tracking model's KL and JS divergence line up quite closely on this particular dataset, so they will simply be referred to as 'divergence'. It can be seen that initially for the comparison with the independent model, it increases, then decreases. There is also a small bubble where it seems to be decreasing for a very short time, but the rate of accumulation increases once again before the final leveling off. This period is much less pronounced in the comparison between experiments for the same model, suggesting that there is a period where some low order statistical characteristics of the models match.

7.2 Interpretability

In conclusion to chapter 4 it was noted that 'model performance' does not necessarily comprise accuracy alone, and interpretability should also be taken into account. In addition to enabling researchers to more fully assess performance and usefulness of a model, interpretability has many other reasons for consideration.

Firstly, we are continuing the march towards more and more ML systems performing safety-critical operations as they tend to outperform humans in certain tasks (Jordan and Mitchell, 2015). One such example is autonomous vehicles. Once we reach full level 5 autonomy (Society of Automotive Engineers, 2014), all control will be taken away from the human driver, to the point where traditional controls such as a steering wheel may no longer even be present in the vehicle. At that stage, the systems controlling these cars will be fully in control of potentially saving human life in accident situations. Even more importantly, on the way to development of that level of technology, we find ourselves in situations where we must give up some control over our safety to developmental software - underpinned of course entirely by ML techniques. It is crucial to interpret and understand models under these conditions, to ensure optimal performance and decrease risk. In fact, many important ethical questions arise in this area regarding the interpretability of the models - for example, what does the model choose to do in a situation where it must decide between preventing an accident but sacrificing an animal; or even saving passenger lives at the cost of sacrificing a pedestrian (Goodall, 2014)? In the case of neural coding, should the technology be advanced sufficiently to be used in applications such as medicine

and mental health screening, similar safety considerations will need to take place.

Another reason for interpretability is to prevent societal discrimination. ML systems are already being utilised in areas that have a large impact on society such as financial planning to decide loan approvals, or targeted advertising for political groups. There is great potential for ML to enhance these by increasing efficiency of capital allocation or engagement. However, there is also a possibility of reinforcing discriminatory practises, which the World Economic Forum is aware of (World Economic Forum, 2018). For example, in a system that decides whether a candidate should be given a job offer, if it has been trained on historical data where women may have been less likely to gain employment, it may capture this feature in the learned model and reinforce it.

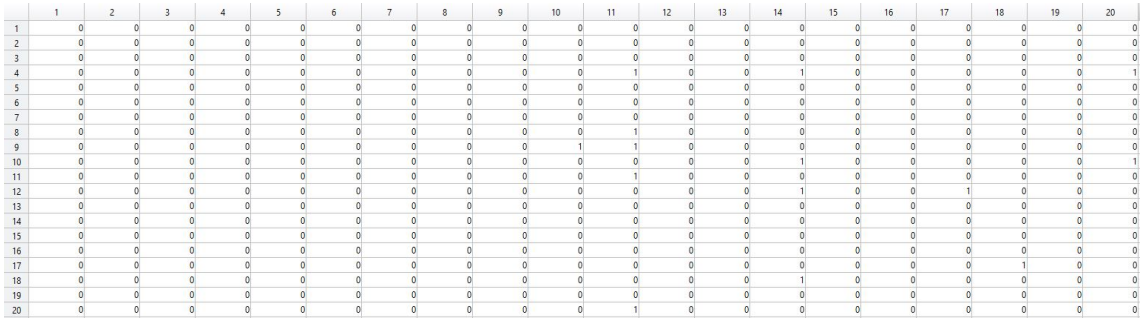
Finally, as the public gains awareness of ML systems being utilised in society, they demand accountability. One example of this is the recently adopted General Data Protection Regulation (GDPR) enacted by the European Union, which sets out new laws on how personal data is to be used by organisations. One impact of this set of laws is that all users now have the right to demand an explanation of how a computerised system has arrived at a decision (Information Commissioner's Office, 2018), for example in the case of an ML system calculating a credit score. It should be clear that without an interpretable model, this explanation is impossible to provide and renders the model illegal. For neural data, the same will be true should these models advance and be used to make any decisions such as choice of appropriate healthcare, or even things such as a subject's most appropriate career based on their neural code screening.

7.2.1 Interpretability Framework

Despite the clear need for interpretability, when surveying the literature it becomes clear that no formal definition for it even exists (Doshi-Velez and Kim, 2017). From a purely linguistic point of view, the definition of interpret is "to explain or tell the meaning of : present in understandable terms" (Merriam-Webster, 2016). Unfortunately the definition is no more concrete than that and leaves many terms open to ambiguity and subjectivity. To what depth does the explanation refer? What is considered 'understandable' depends on who you are. From an ML standpoint, we can attempt to extend this nebulous definition to say that an interpretable model is one that models the underlying system in a way that is understandable and explainable to humans.

From this emerges two rough classes of interpretability. The first is analytical and in what literature exists, it is explored well (Vellido et al., 2012). This deals with quantifiable aspects, such as scale and dimensionality of the model, or number of parameters. This satisfied the above description because numbers and mathematical representations are familiar to humans, especially to researchers, and provide a well-constrained framework within which to compare interpretability. The second class of interpretability is more of an interactive form, satisfying the subjective aspect of the above definition. This deals with the capacity of the model to allow an experimenter to alter the model and observe that alteration's direct effects. One example of this could be simply manually adjusting hyperparameters and measuring the change in results on the fly, meaning the model's hyperparameters have to map to its output in a fairly direct and understandable way. Another example of this is being explored by Morcos et al. (2018). In this research, nodes of a neural network are removed entirely to measure the impact of subsystems of various size's impact on the output, giving an indication on the form of the solution the model has actually learned.

Some take the approach that machine learning systems can be interpretable by output



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 7.1: A subsection of a $164x100,000$ sample from a learned distribution (over the Allen Institute dataset) according to the population tracking model. A neuron is predicted to spike, given the state of the other neurons and the state of the population rate, when denoted by a 1. The sample demonstrates the sparsity of the data.

alone (Vellido et al., 2012). There are many ML systems that generate outputs without human intervention and are trusted in mission-critical systems such as aeroplane crash-avoidance or financial stock market purchasing. It is argued that the very act of them producing a useful, actionable output makes them interpretable indirectly through this output. The advent of more complex systems such as those resulting from the deep learning paradigm renders this less true. These systems can produce outputs that are very transformed compared to the original data having been processed through dozens of layers of nodes. Therefore a simple output is no longer enough to consider a model interpretable. Overall, a rigorous approach is very much still lacking. Abdul et al. (2018) attempts to take a data-driven approach to interpretability analysis, analysing 289 papers as well as 12,412 citations to extract trends in explainability and identify connected features of interpretable models, in an attempt to gather features that may define more rigorously an 'interpretable' model. Some interesting insights from this study include the identification that methods to explain earlier ML systems do not share many characteristics with those used to explain more modern ones, suggesting the new approaches are using very different techniques; and differences between models relying on synthetic data when compared with models trained entirely on real-world data.

With the definitions, pitfalls, and current literature in in mind, particularly those of the review paper by Vellido et al. (2012) some criteria may be extracted, summarised in table 7.1. The first three criteria fall within the analytical class of interpretability rules, whilst the last three are more subjective but try to capture the subjective aspect of the definition.

7.2.2 Interpretability Comparison

Given the criteria in section 7.2.1, table 7.2 provides a detailed comparison of some of the models discussed and executed as part of this project. Of course subjectivity plays a role in this comparison as has been made clear above, but it nonetheless represents a step on the way to a rigorous understanding of interpretability as an inherent property and performance metric of ML systems.

7.3 Conclusion

Criteria	Less Interpretable	More Interpretable
Num. of Parameters	Many params e.g. N^2	Few params e.g. 2^N
Variable Dimensionality	High dimensionality - too many features	Low dimensionality - too few features
Dimensionality Reduction (DR)	Large dimensionality compression	Remains similar in dimension to input data
Ease of Visualisation	Graphical form shares few characteristics with learned model	Easy to represent in graphical form
Human-Imposed (HI) Structure	Less structure - less constrained	More structure - more constrained
User-Friendliness	Harder to adjust hyperparams, slow iteration speed for experimenting with model, etc.	Easier to adjust hyperparams, fast iteration speed for experimenting with model, etc.

Table 7.1: Criteria for an interpretability comparison framework.

Model	**# Params**	**Dimensionality**	**DR**	**Ease of Vis.**	**HI Structure**	**User-Friendliness**
Full Distribution	2^N	Low	None	Poor	Low	Medium
Pairwise ME	N^2	Medium	None	Poor	Low-Medium	Medium-High
SVM	Data-Dependent (Low-Medium)	Low	High	High	Low	Poor-Fair
Population Tracking	N^2	Low	Low	Poor	Low-Medium	High
RBM	Hyperparam-Dependent $\frac{n(n+1)}{2}$	Medium	Low	Low	Low	Poor

Table 7.2: A subjective comparison of the primary models explored in this paper, grounded in the interpretability framework defined in section 7.2.1

.

Chapter 8

Conclusion

8.1 Summary of Findings

This project exists to assess the current state of research into unraveling the neural code. It aimed to outline the role computational methods can play in advancing that research, and discover how appropriate they are to specific sub-problem domains. One of its purposes was to discover existing techniques for characterising the neural code. It sought to compare these models based on several factors; primarily accuracy, computational cost, and interpretability, to assess the current state of ML applied to the field of neural coding.

The background of the project was explored in depth. In chapter two the neural code was introduced; the brain's internal mapping between stimulus and brain activity. A large area of research in neuroscience that is working towards understanding the neural code was explored. A number of complexities that make neural coding a difficult domain came to light. Given the complexity of the domain, the literature in chapter 3 revealed how researchers have turned to computational methods to aid them. In particular, techniques within the field of machine learning (ML) are ripe for application to this kind of data. As defined in the chapter, ML is capable of extracting knowledge from large data sets without being explicitly programmed to do so. Many papers are being published around the subject, and a representative sample of the approaches being taken was provided in chapter 4. This led to an extensive list of types of model. Throughout the survey, it emerged that the issue which receives most attention is how to model population neuron activity at scale as imaging technology advances. Few papers made much mention of interpretability of their models. In fact, the only time this point appeared was on one brief occasion when discussing maximum entropy models. This exposed a huge gap in the body of current research; we must be able to identify why a model works and why it is the best amongst all possible models, in order to ensure we have the best method available.

Moving on to the execution phase, a comprehensive data science pipeline was introduced in-depth in chapter 5. A description of the steps taken, issues faced, and technical features of this pipeline provided insight into how one embarks on the analysis of neural data. A number of models were built, involving linear decoders (both of which emerged similarly in terms of performance), and models constructed by the population tracking model, related to entropy and probabilistic in nature. These two classes of algorithm gave good insight into the two main approaches into neural modeling highlighted in chapter 4; those based on internal dynamics and those based on stimulus. How these two techniques performed was assessed and compared in the following two chapters.

Overall it seems the current state of neural coding is maturing, but still has a long way

to go. The range of models attests to this, as a smaller subset of examples has not had the time to prove themselves and outshine the rest. Whilst some techniques show good accuracy, they tend to come with limitations and in many cases it remains unclear if they will scale to even larger populations of neurons as imaging technology improves and the datasets available become larger. project has achieved its goal of generating an extensive list of motivations, approaches, and considerations.

With the above under consideration, this project's main contribution has been to improve neural code modeling by selecting amongst the vast range of methods and assessing their applicability to a real-world, large-scale dataset with considerable values of N. It has also attempted to provide a framework to characterise the broader model of the neural code by placing a significant focus on interpretability of the models throughout; right from the research stage through to execution and analysis. It has also contributed comments on the form of the neural code to extend previous work, such as the identification of differing performance for different brain conditions and parameters, as well as differing reproduced population distributions for various different experiment types.

8.2 Project Evaluation

Now the project is complete, it is useful to revisit the metrics defined in chapter 1, to see if the project as it has been planned is able to satisfy them.

8.2.1 Objectives

1. *Use existing machine learning (ML) code and implement further ML techniques if required, to analyse brain activity data from mice.* A range of techniques were identified in chapter 4. These were explored and implemented in chapter 5.

2. *Utilise statistical tools to decode the stimulus and infer a model for the response.* The models that facilitated this statistical analysis were utilised in 5 and explored further in chapter 6.

3. *Compare the efficacy of methods and models used.* A comparison framework was devised in chapter 7, which the models/results were then compared against.

4. *Investigate the learned model to examine the form of the neural code itself.* The interpretability was considered throughout the project, and particularly highlighted in chapter 7. Conclusions were drawn around the form of the neural code in 8.

8.2.2 Deliverables

- *Literature review of the state of the art in the field, to contextualise and focus the rest of the work in the project.* An extensive review of the background and state of the art was delivered across chapters 2, 3, and 4.

- *Implementation followed by quantitative analysis and comparison of various models' strengths and weaknesses.* Use of existing models/further code was implemented in 5, and a detailed comparison was undertaken in chapter 7.

- *Qualitative theoretical analysis required to gain insight into the neural code - drawing wider conclusions about its structure, to produce some insight into the encoding for*

further work to build upon. Interpretation and analysis was continued throughout chapters 7 and 8.

8.2.3 Added Value

The added value has been brought by the above satisfied deliverables and objectives. This added value was specified as:

- Intends to play its part in contributing to data analysis of the massive amount of data
- Analysis of a novel dataset
- Be part of a greater march towards more comprehensive sampling and analysis of the brain to generalise across other brain regions

8.3 Limitations & Future Work

The main limitation of this study is the number of models covered. The RBM was identified as a model to explore in section 5.2 however due to time constraints only the initial steps were taken to implement it, and no analysis was undertaken. This issue is inherent due to the sheer number of models available, many of which have significant crossover in terms of applicability. In general, the research papers that propose these models only compare performance based on a 'ground truth', rather than against other common models. This makes it difficult to select amongst them without actually implementing them, and therefore the number available can appear overwhelming.

Another limitation is the analysis of the models in terms of the form of the neural code. Due to the lack of research regarding interpretation, many of these models do not have good frameworks with which to assess or monitor interpretability. Some of them do have some features which contribute to class 2 interpretability - i.e. features that enable the user to explore and alter the model. An example of this is being able to sample directly from learned models in the population tracking model (ODonnell et al., 2016), but especially in terms of analytical class 1 interpretability they are severely lacking.

The results of the linear decoder accuracies for the shuffled dataset was somewhat unexpected, in the performance was significantly worse - even though other contemporary research suggests this should not be the case. Extensions to this project should look into why this was the case.

One of the primary limitations of the field in general is that despite advances in imaging technology, populations are still far too small. It has been established that the brain has around 85 billion neurons (Azevedo et al., 2009), but the most advanced laboratories are only recording at low temporal resolution from a few thousand neurons (Stevenson and Kording, 2011). This begs the question: how much can such a small percentage of neurons really be contributing to the the neural code as a whole? Even if the technology did exist, it has been suggested after analysis of the predicted number of neurons present, that current methods would be unable to record most neurons, either due to sheer density or because most neurons remain silent most of the time (sparsity) (Pedreira et al., 2012). There needs to be significant advancements in recording technology to enable researchers to truly delve into the neural code. This is not to say time spent developing modeling techniques on smaller populations is time wasted, as the principles learned will surely be

useful at scale; but perhaps the expectation for the research to really uncover anything groundbreaking with tiny neural populations should be laid to one side for now.

A planned item of future work is to port the implementation of the population tracking method to Python. Initially this was intended but could not be completed due to time constraints. This would avoid the technical hurdles faced during the data pipeline, and would be particularly beneficial for the Allen Institute dataset since the `allensdk` is written in Python, and would therefore reduce development friction when working with this dataset. Another planned future work is to extend the paper with further models from varying approaches to provide a more comprehensive analysis and an overall assessment of which is best suited to this very exciting dataset in particular.

8.4 Personal Evaluation

In terms of my personal development throughout the project, the project has been a fascinating experience. It is a pleasure to be able to dive into a large-scale project in-depth, working with real data and real, applicable models; and is something I have not experienced before.

The first major skill that has been enhanced through the undertaking of this project is my research skills. Whilst I have conducted literature surveys before and critically analysed research papers, both the material itself was not nearly as advanced as that covered in this project, and I believe the quality of my research was not as good. Once I realised the scale of the research that would need to be undertaken, a systematic approach to the review was taken. Firstly I identified 3 or 4 key papers with extremely high reference counts in the field. These were read, and shared citations, interesting topics, or common authors were noted to provide the papers for the next cycle of the review. This process was repeated iteratively until a strong picture of leading authors and publications had been constructed, and for any further research I may do in the field, I believe this has process has set me up excellently to quickly get a good overview of the current state of research at any time. Following identification of papers, a framework was set out for analysis with a methodology for reading the papers, as well as specific questions and categories of notes to be recorded were pre-planned and filled in as I went through the paper, producing a high level of analysis for each of them.

The second major skill I have learnt is end-to-end data analysis over a large dataset. It was a pleasure to work through acquisition, pre-processing, tuning, model construction and performance analysis; it was particularly satisfying that the dataset was a recent, real-world set with valuable insights to reveal. Just how difficult it can be to wrangle data into the correct form became apparent, involving accounting for missing values, jagged matrices, massive datasets impossible to fit in memory, and other challenges. I believe this is a skill that will suit me well going forward into further research or industry.

Part IV

Appendices

What is the Neural Code?

Adam Matheson
MSc Advanced Computing - Machine Learning/Data Mining
Supervisor: Dr. Cian O'Donnell

University of BRISTOL

1. Intro/Background

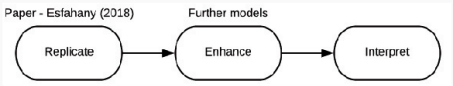
Neuroscience
- How does the brain internally represent information?
- Phsyiological mechanisms understood - via patterns of electrical activity in neurons
- But *why* does a particular stimuli result in a particular pattern of activity in the brain?
- The mapping between sensory stimuli => brain activity is the neural code.

Machine Learning (ML)
- Algorithmically discover patterns in data
- Exploit these patterns to take action - e.g. classify[1]
- Enhances human learning with speed and capacity

Let's merge these to exploit discovered patterns, and learn more about the inner workings of the brain!

2. Methodology

3 Phases: Replicate[2] => Enhance => Interpret



Programming Language: Python
- Large package ecosystem (especially for ML), easy to do data manipulation, large support community
- Poor performance at times - but iteration time is more important than execution time in this context

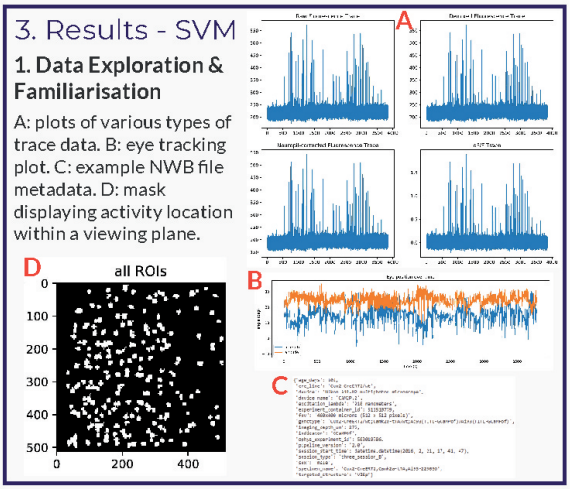
Packages
- allensdk[3]: accessing data/metadata & cell metrics
- NumPy: mathematical operations, arrays/matrices
- Matplotlib: pretty graphs, figures, etc.
- Pandas: flexible, versatile DataFrame data structure
- SciPy stats: statistical operations
- scikit-learn: data splitting, off-the-shelf ML algorithms, accuracy scoring & comparison

Models
- Linear Regression & Support Vector Machine
- Independent, population tracking[4]
- Pairwise maximum entropy (exact, MCMC)[5]
- Restricted Boltzmann machine[6]

3. Results - SVM

1. Data Exploration & Familiarisation

A: plots of various types of trace data. B: eye tracking plot. C: example NWB file metadata. D: mask displaying activity location within a viewing plane.



2. Get Data

3. Feature Extraction
- 3 hours of traces, split into stimulus epochs
- Segment each epoch into 10s intervals
- Compute mean dF/F per interval, label the stimulus
- Z-score, combine => feature (12,123,614 data points)

4. Split Data
- Subsample without replacement an equal amount of data per class
- Subsample size = 80% of the smallest class size
- Test set consists of remaining samples, unbalanced

5. Optimise L2 Regularisation via X-Validation
- Train & test model for each constant

$$C = \{10^{-2}, 10^{-1}, 1, 10, 10^{2}, 10^{3}, \infty\}$$

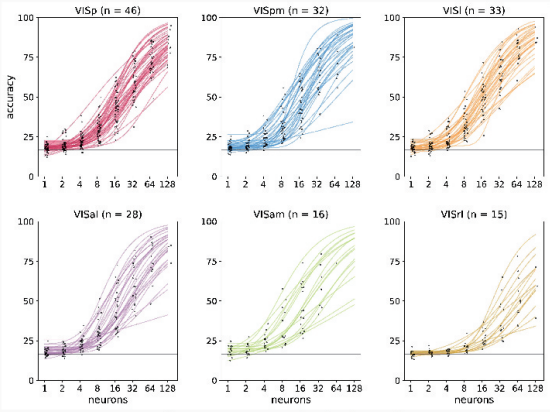
- Make a prediction, record accuracy
- Keep C value corresponding to most accurate model

6. Construct Models
- Randomly resample neurons without replacement to investigate accuracy as a function of population size

$$n = \{2^{0}, 2^{1}, 2^{2}, 2^{3}, 2^{4}, 2^{5}, 2^{6}, 2^{7}\}$$

7. Assess Accuracy
- Extrapolate accuracy as function of n with generalised logistic curve:

$$\text{accuracy}(n) = \frac{1-c}{(1+e^{-an})^{b}} + c$$



Above: accuracy by brain region

4. Conclusions/Todo

- Constant trade-off between accuracy & tractability
- Investigate further models: independent makes large assumptions but seems to perform well; population tracking looks promising; pairwise ME may not be scalable
- Interpretability is neglected - essential to assess params etc. to understand models and draw wider conclusions

5. References

[1] Bishop, C. (2006). Pattern recognition and machine learning. New York: Springer.
[2] Esfahany, K., Siergiej, I., Zhao, Y. and Park, I. (2018). Organization of Neural Population Code in Mouse Visual System. Society for Neuroscience, 5(4).
[3] Allen Institute for Brain Science (2015). Allen Brain Atlas API. Available from: brain-map.org/api/index.html
[4] O'Donnell, C., Gonçalves, J.T., Whiteley, N., Portera-Cailliau, C. and Sejnowski, T.J., 2017. The population tracking model: a simple, scalable statistical model for neural population data. Neural computation, 29(1), pp.50-93.
[5] G Schwartz, JH Macke, D Amodei, H Tang, MJ Berry: Low error discrimination using a correlated population code. Journal of Neurophysiology, 108(4), 1069-1088, 04 2012
[6] J Sohl-Dickstein, P Battaglino, MR DeWeese (2011). Minimum probability flow learning. International Conference on Machine Learning

Bibliography

Abdul, A., Vermeulen, J., Wang, D., Lim, B. Y., and Kankanhalli, M. (2018). Trends and Trajectories for Explainable, Accountable and Intelligible Systems: An HCI Research Agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 582:1–582:18, New York, NY, USA. ACM.

Adrian, E. D. and Zotterman, Y. (1926). The impulses produced by sensory nerve endings. *The Journal of Physiology*, 61(4):465–483.

Allen Brain Institute (2017a). Stimulus Set and Response Analysis.

Allen Brain Institute (2017b). Visual Coding Overview.

Allen Brain Institute (2018). Allensdk github. original-date: 2015-05-07T18:35:56Z.

Azevedo, F. A. C., Carvalho, L. R. B., Grinberg, L. T., Farfel, J. M., Ferretti, R. E. L., Leite, R. E. P., Filho, W. J., Lent, R., and HerculanoHouzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541.

Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6.

Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.*, 6(1):20–29.

Berens, P., Freeman, J., Deneux, T., Chenkov, N., McColgan, T., Speiser, A., Macke, J. H., Turaga, S. C., Mineault, P., Rupprecht, P., Gerhard, S., Friedrich, R. W., Friedrich, J., Paninski, L., Pachitariu, M., Harris, K. D., Bolte, B., Machado, T. A., Ringach, D., Stone, J., Rogerson, L. E., Sofroniew, N. J., Reimer, J., Froudarakis, E., Euler, T., Rosn, M. R., Theis, L., Tolias, A. S., and Bethge, M. (2018). Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLOS Computational Biology*, 14(5):e1006157.

Biederman (1987). Recognition-by-components - a theory of human image understanding.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. Google-Books-ID: kTNoQgAACAAJ.

Bohte, S. M., Spekreijse, H., and Roelfsema, P. R. (2000). The Effects of Pair-wise and Higher-order Correlations on the Firing Rate of a Postsynaptic Neuron. *Neural Computation*, 12(1):153–179.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Breakspear, M. (2017). Dynamic models of large-scale brain activity. *Nature Neuroscience*, 20(3):340–352.

Breiman, L. (2017). *Classification and Regression Trees*. Routledge.

Brown, E. N., Kass, R. E., and Mitra, P. P. (2004). Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7(5):456–461.

Davatzikos, C., Ruparel, K., Fan, Y., Shen, D., Acharyya, M., Loughead, J., Gur, R., and Langleben, D. (2005). Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection. *NeuroImage*, 28(3):663–668.

Dayan, P. and Abbott, L. F. (2001). *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational neuroscience. Massachusetts Institute of Technology Press, Cambridge, Mass.

Doshi-Velez, F. and Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *arXiv:1702.08608 [cs, stat]*. arXiv: 1702.08608.

Esfahany, K., Siergiej, I., Zhao, Y., and Park, I. M. (2017). Organization of Neural Population Code in Mouse Visual System. *bioRxiv*, page 220558.

Fuhs, M. C. and Touretzky, D. S. (2006). A Spin Glass Model of Path Integration in Rat Medial Entorhinal Cortex. *Journal of Neuroscience*, 26(16):4266–4276.

Ganmor, E., Segev, R., and Schneidman, E. (2011). Sparse low-order interaction network underlies a highly correlated and learnable neural population code. *Proceedings of the National Academy of Sciences*, 108(23):9679–9684.

Gao, Y., Archer, E., Paninski, L., and Cunningham, J. P. (2016). Linear dynamical neural population models through nonlinear embeddings. *arXiv:1605.08454 [q-bio, stat]*. arXiv: 1605.08454.

Gerwinn, S., Berens, P., and Bethge, M. (2009). A joint maximum-entropy model for binary neural population patterns and continuous signals. page 9.

Goodall, N. J. (2014). Ethical Decision Making during Automated Vehicle Crashes. *Transportation Research Record*, 2424(1):58–65.

Granot-Atedgi, E., Tkaik, G., Segev, R., and Schneidman, E. (2013). Stimulus-dependent Maximum Entropy Models of Neural Population Codes. *PLoS Computational Biology*, 9(3):e1002922.

Gross, G. W., Rieske, E., Kreutzberg, G. W., and Meyer, A. (1977). A new fixed-array multi-microelectrode system designed for long-term monitoring of extracellular single unit neuronal activity in vitro. *Neuroscience Letters*, 6(2):101–105.

Hamill, O. P., Marty, A., Neher, E., Sakmann, B., and Sigworth, F. J. (1981). Improved patch-clamp techniques for high-resolution current recording from cells and cell-free membrane patches. *Pflgers Archiv*, 391(2):85–100.

Hodgkin, A. and Huxley, A. (1939). Action Potentials Recorded from Inside a Nerve Fibre. *Nature*, 144:710–711.

Humplik, J. and Tkaik, G. (2016). Semiparametric energy-based probabilistic models. *arXiv:1605.07371 [cond-mat, q-bio, stat]*. arXiv: 1605.07371.

Hunter, J. D. (2007). Matplotlib: A 2d Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95.

Idreos, S., Papaemmanouil, O., and Chaudhuri, S. (2015). Overview of Data Exploration Techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 277–281, New York, NY, USA. ACM.

Information Commissioner's Office (2018). Rights related to automated decision making including profiling.

Izhikevich, E. M. (2004). Spike-timing Dynamics of Neuronal Groups. *Cerebral Cortex*, 14(8):933–944.

Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *Physical Review*, 106(4):620–630.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python. [Online; accessed ¡today¿].

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Jorgenson, L. A., Newsome, W. T., Anderson, D. J., Bargmann, C. I., Brown, E. N., Deisseroth, K., Donoghue, J. P., Hudson, K. L., Ling, G. S. F., MacLeish, P. R., Marder, E., Normann, R. A., Sanes, J. R., Schnitzer, M. J., Sejnowski, T. J., Tank, D. W., Tsien, R. Y., Ugurbil, K., and Wingfield, J. C. (2015). The BRAIN Initiative: developing technology to catalyse neuroscience discovery. *Phil. Trans. R. Soc. B*, 370(1668):20140164.

Kamitani, Y. and Tong, F. (2005). Decoding the visual and subjective contents of the human brain. *Nature neuroscience*, 8(5):679–685.

Kluyver, T., Ragan-Kelley, B., Prez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and Unknown], J. d. t. (2016). Jupyter Notebooks a publishing format for reproducible computational workflows. In Loizides, F. and Scmidt, B., editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press.

Knill, D. C. and Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719.

Kster, U., Sohl-Dickstein, J., Gray, C. M., and Olshausen, B. A. (2014). Modeling Higher-Order Correlations within Cortical Microcolumns. *PLOS Computational Biology*, 10(7):e1003684.

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

Lewicki, M. S. (1998). A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78.

Li, M. and Tsien, J. Z. (2017). Neural CodeNeural Self-information Theory on How Cell-Assembly Code Rises from Spike Time and Neuronal Variability. *Frontiers in Cellular Neuroscience*, 11.

Logothetis, N. K., Pauls, J., Augath, M., Trinath, T., and Oeltermann, A. (2001). Neurophysiological investigation of the basis of the fMRI signal. *Nature*, 412(6843):150–157.

Martignon, L., Von Hassein, H., Grn, S., Aertsen, A., and Palm, G. (1995). Detecting higher-order interactions among the spiking events in a group of neurons. *Biological Cybernetics*, 73(1):69–81.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. pages 51–56.

Meister, M. and Berry, M. J. (1999). The Neural Code of the Retina. *Neuron*, 22(3):435–450.

Merriam-Webster (2016). *The Merriam-Webster Dictionary*. Merriam-Webster, Incorporated.

Meshulam, L., Gauthier, J. L., Brody, C. D., Tank, D. W., and Bialek, W. (2017). Collective Behavior of Place and Non-place Neurons in the Hippocampal Network. *Neuron*, 96(5):1178–1191.e4.

Morcos, A. S., Barrett, D. G. T., Rabinowitz, N. C., and Botvinick, M. (2018). On the importance of single directions for generalization. *arXiv:1803.06959 [cs, stat]*. arXiv: 1803.06959.

Norman, K. A., Polyn, S. M., Detre, G. J., and Haxby, J. V. (2006). Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9):424–430.

Okun, M., Steinmetz, N. A., Cossell, L., Iacaruso, M. F., Ko, H., Barth, P., Moore, T., Hofer, S. B., Mrsic-Flogel, T. D., Carandini, M., and Harris, K. D. (2015). Diverse coupling of neurons to populations in sensory cortex. *Nature*, 521(7553):511–515.

Oliphant, T. E. (2015). *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition.

ODonnell, C., Gonalves, J. T., Whiteley, N., Portera-Cailliau, C., and Sejnowski, T. J. (2016). The Population Tracking Model: A Simple, Scalable Statistical Model for Neural Population Data. *Neural Computation*, 29(1):50–93.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pedreira, C., Martinez, J., Ison, M. J., and Quian Quiroga, R. (2012). How many neurons can we see with current spike sorting algorithms? *Journal of Neuroscience Methods*, 211(1):58–65.

Pereira, F., Mitchell, T., and Botvinick, M. (2009). Machine learning classifiers and fMRI: A tutorial overview. *NeuroImage*, 45(1):S199–S209.

Pillow, J. W., Paninski, L., Uzzell, V. J., Simoncelli, E. P., and Chichilnisky, E. J. (2005). Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 25(47):11003–11013.

Polyn, S. M., Natu, V. S., Cohen, J. D., and Norman, K. A. (2005). Category-Specific Cortical Activity Precedes Retrieval During Memory Search. *Science*, 310(5756):1963–1966.

Richmond, B. J. (2009). Information Coding. In Squire, L. R., editor, *Encyclopedia of Neuroscience*, pages 137–144. Academic Press, Oxford.

Riess, A. G., Filippenko, A. V., Challis, P., Clocchiatti, A., Diercks, A., Garnavich, P. M., Gilliland, R. L., Hogan, C. J., Jha, S., Kirshner, R. P., Leibundgut, B., Phillips, M. M., Reiss, D., Schmidt, B. P., Schommer, R. A., Smith, R. C., Spyromilio, J., Stubbs, C., Suntzeff, N. B., and Tonry, J. (1998). Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant. *The Astronomical Journal*, 116(3):1009–1038.

Russell, J. T. (2011). Imaging calcium signals *in vivo* : a powerful tool in physiology and pharmacology: *In vivo* calcium imaging. *British Journal of Pharmacology*, 163(8):1605–1625.

Santos, G. S., Gireesh, E. D., Plenz, D., and Nakahara, H. (2010). Hierarchical interaction structure of neural activities in cortical slice cultures. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 30(26):8720–8733.

Schacter, D. L., Reiman, E., Uecker, A., Roister, M. R., Yun, L. S., and Cooper, L. A. (1995). Brain regions associated with retrieval of structurally coherent visual information. *Nature*, 376(6541):587–590.

Schaub, M. T. and Schultz, S. R. (2012). The Ising decoder: reading out the activity of large neural ensembles. *Journal of Computational Neuroscience*, 32(1):101–118.

Schneidman, E., Berry, M. J., Segev, R., and Bialek, W. (2006). Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012.

Schwartz, G., Macke, J., Amodei, D., Tang, H., and Berry, M. J. (2012). Low error discrimination using a correlated population code. *Journal of Neurophysiology*, 108(4):1069–1088.

Society of Automotive Engineers (2014). SAE International Standard J3016.

Stein, R. B., Gossen, E. R., and Jones, K. E. (2005). Neuronal variability: noise or part of the signal? *Nature Reviews Neuroscience*, 6(5):389–397.

Steinmetz, N. A., Buetfering, C., Lecoq, J., Lee, C. R., Peters, A. J., Jacobs, E. A. K., Coen, P., Ollerenshaw, D. R., Valley, M. T., de Vries, S. E. J., Garrett, M., Zhuang, J., Groblewski, P. A., Manavi, S., Miles, J., White, C., Lee, E., Griffin, F., Larkin, J. D., Roll, K., Cross, S., Nguyen, T. V., Larsen, R., Pendergraft, J., Daigle, T., Tasic, B., Thompson, C. L., Waters, J., Olsen, S., Margolis, D. J., Zeng, H., Hausser, M., Carandini, M., and Harris, K. D. (2017). Aberrant Cortical Activity in Multiple GCaMP6-Expressing Transgenic Mouse Lines. *eNeuro*, 4(5).

Stephen Michael Kosslyn (1975). Information Representation in Visual Images. *Cognitive Psychology*, 7:341–370.

Stevenson, I. H. and Kording, K. P. (2011). How advances in neural recording affect data analysis. *Nature Neuroscience*, 14(2):139–142.

Theis, L., Berens, P., Froudarakis, E., Reimer, J., RomnRosn, M., Baden, T., Euler, T., Tolias, A., and Bethge, M. (2016). Benchmarking Spike Rate Inference in Population Calcium Imaging. *Neuron*, 90(3):471–482.

Tkaik, G., Marre, O., Amodei, D., Schneidman, E., Bialek, W., and Ii, M. J. B. (2014). Searching for Collective Behavior in a Large Network of Sensory Neurons. *PLOS Computational Biology*, 10(1):e1003408.

Tkaik, G., Marre, O., Mora, T., Amodei, D., Berry II, M. J., and Bialek, W. (2013). The simplest maximum entropy model for collective behavior in a neural network. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(03):P03011. arXiv: 1207.6319.

Tkaik, G., Schneidman, E., Berry II, M. J., and Bialek, W. (2006). Ising models for networks of real neurons. *arXiv:q-bio/0611072*. arXiv: q-bio/0611072.

Uffink, J. (1997). The Constraint Rule of the Maximum Entropy Principle. *Studies in History and Philosophy of Science Part B - Studies in History and Philosophy of Modern Physics*, 27.

Vellido, A., Martn-Guerrero, J. D., and Lisboa, P. J. G. (2012). Making machine learning models interpretable. In *In Proc. European Symposium on Artificial Neural Networks, Computational Inligence and Machine Learning*.

Weiss, S. M. and Kulikowski, C. A. (1991). *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Werner, G. and Mountcastle, V. B. (1963). The variability of central neural activity in a sensory system, and its implications for the central reflection of sensory events. *Journal of Neurophysiology*, 26:958–977.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann. Google-Books-ID: 1SylCgAAQBAJ.

World Economic Forum (2018). How to Prevent Discriminatory Outcomes in Machine Learning.

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624.