Learning interpretable theories for complex neural systems

Alexandre René

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
Doctorate of Philosophy in Physics

Ottawa-Carleton Institute of Physics
Department of Physics
Faculty of Science
University of Ottawa

Abstract

Mathematical modelling has a long history of providing us with useful, succinct descriptions of the natural world. More than mere descriptions, the resulting models provide predictions that can be tested within the framework of the scientific method. But as we seek however to understand progressively more complex systems—especially in interdisciplinary fields like neurophysics—the methods of old don't always yield the models we want: often they are too crude, or have too many free parameters—either of which makes them difficult to test.

This work develops new paths to modelling, meeting complexity with today's abundance of computational power. In contrast to other work however, which may outright replace the models of old with highly accurate but inscrutable ones, such as neural networks, here we stay firmly in the modelling tradition. By focussing on mechanistic, equation-based models, we extend rather than replace existing modelling capabilities.

First we develop methods to *learn* complex models, where the form of the equations are known from our understanding of biology, but the effective parameters can only be inferred by fitting the entire model to data. These methods further allow to characterize correlations between parameters using Markov chain Monte Carlo (MCMC). However they can also lead to the proliferation of candidate models, which all plausibly fit the data. Therefore in a second part we develop a new approach to model *comparison*, integrating the principles of scientific induction with the methods of machine learning. The method is based on a novel random walk on the cumulative distribution function of errors; we thereby obtain an empirical modelling discrepancy (EMD) statistic that accounts for epistemic uncertainty on the model, specifically uncertainty under replications of the experiment.

Together, these methods can be seen as the basis of a program for expert-driven, computer-enhanced research. And while developed in the context of computational neuroscience, they should be applicable wherever one is concerned with designing and studying mechanistic models, be they models in computational biology, statistical physics, or even econometrics.

Résumé

Les techniques de modélisation mathématique nous ont longtemps fournis des descriptions utiles et succinctes du monde naturel. Plus que de simple descriptions, les modèles ainsi obtenus fournissent des prédictions pouvant être validées à l'intérieur du cadre de la méthode scientifique. En cherchant toutefois à comprendre des systèmes de plus on plus complexes—en particulier dans les domaines interdisciplinaires comme la neurophysique—ces méthodes arrivent parfois à leurs limites: les modèles sont trop simplistes, ou possèdent trop de paramètres indéterminés—et par conséquent peuvent rarement être validés.

Ce travail développe de nouveaux moyens de modélisation, répondant à la complexité avec l'abundance de puissance de calcul dont on bénéficie aujourd'hui. À l'inverse d'autres efforts toutefois, il n'est pas question ici de remplacer les modèles conventionnels par d'autres plus précis mais indéchiffrables tel que des réseaux neuronaux. En se concentrant sur des modèles mécanistiques, exprimés à l'aide d'équations, on reste plutôt fermement ancré dans la tradition de modélisation, étendant ses possibilités au lieu de les remplacer.

On développe d'abord des méthodes pour *apprendre* des modèles, où la forme des équations découle de notre connaissance biologique, mais où les paramètres doivent être inférés en adaptant le modèle entier à des données observées. Une fois les paramètres appris, ces méthodes permettent ensuite également de caractériser leurs corrélations à l'aide de Monte-Carlo par chaînes de Markov (MCMC). Toutefois, elles peuvent aussi mener à un prolifération de modèles candidats, chacun ressemblant aux données. C'est pourquoi en seconde partie on développe une nouvelle perspective sur la sélection de modèles, intégrant les principes d'induction scientifique avec les méthodes d'apprentissage automatique. L'approache est basée sur un nouveau processus stochastique sur la distribution cumulative des erreurs; on obtient ainsi une statistique empirique de la divergence du modèle (qu'on nomme EMD, pour *empirical modelling discrepancy*). Cette statistique tient compte de l'incertitude épistemique du modèle, spécifiquement l'incertitude due aux réplications de l'expérience.

Ensemble, ces méthodes peuvent être vues comme formant la base d'un programme de recherche mené par les experts et renforcé par le calcul. Et bien que qu'elles aient été développées dans le contexte de la neuroscience computationelle, elles pourraient être appliquées à tout contexte où l'on conçoit et étudie des modèles mécanistiques—comme la biologie computationelle, la physique statistique, ou même l'économétrie.

Acknowledgments

First and foremost I wish to thank my supervisor, Prof. André Longtin, for his unflinching support throughout this journey. His willingness to let me explore new directions, and patience to see where they pan out, has brought us to scientific corners neither of us would have, and this work is the richer for it. And his continued insistence that all things always be explained clearly and simply have ensured that these ideas are now not only in my head, but accessible to the public.

I also owe an immense debt to Prof. Jakob Macke. Not only for hosting me in his research group at the caesar in Bonn during the initial phase of my PhD, but for setting me on this research path. It is safe to say that he has had an influence on every one of the following chapters, either directly or indirectly.

I must also thank my hosts at Forschungszentrum Jülich and RWTH Aachen, in particular Prof. Moritz Helias, for intellectual and material support as I transitioned into post-doctoral employment. Although much of the work we did together falls outside the scope of this thesis, our collaboration undoubtedly shaped the later chapters, and has without a doubt forever shaped my understanding of deep neural networks.

Finally, it would be unconscionable for me not to mention my partner, Marie-Claude Dicaire, to whom I owe innumerable evenings and weekends as I worked on these pages. She has been an unflinching support over the years, and I can only hope to make it up to her in the future.

Contents

List of Figures

List of Tables

Introduction

Introduction

The natural world is filled with by fascinating phenomena at all scales, which it is our role and passion as scientists to explain. Unfortunately, the tool we ultimately rely on to do this—our mind—is frustratingly limited: it likes simple stories and models, and can only hold a handful of items in working memory at once.

This is why, even after centuries of scientific progress, our greatest explanatory successes occur when we describe a complicated system in terms of a few numbers or *interactions*:

- an orbit in terms of inclination and eccentricity;
- the spread of a disease in terms of its reproduction ratio;
- a gas in terms of its pressure, volume and temperature;
- a chemical reaction in terms of its kinetic constants;
- electron orbitals in terms of a few quantum numbers;
- electron screening in terms of an effective charge;
- a dynamical system in terms of fixed point and limit cycles;
- or more generally, any classical system in terms of its potential and kinetic energies.

These numbers rely on identifying symmetries in a system, and they allow us to construct simple stories that fit into a 45 minute talk, a half dozen figures, or a 300 word abstract. In more complex models, often these symmetries are only approximate, and correctly deriving the appropriate effective model requires considerable work and expertise.

But what should we do when there are no such symmetries, or the task of finding them too formidable?

My own research concerns models of neural dynamics, for which such questions are unavoidable: with trillions of interactions between neurons, the understanding of neural systems would seem to require statistical methods. And yet, those interactions are collectively and heterogeneously tuned to produce very specific patterns—which any statistical averaging is likely to destroy. To do this correctly, we would need to identify symmetries in spaces with thousands of dimensions or more. This is beyond the capacity of the human mind, but with the aid of computers we can scale our methods to high-dimensional spaces. Of course computers still need to be programmed, so while they are adept at testing theories and increasing scale, they are of little help to derive new models.

Or so it seemed. With the advent of neural networks trained with deep learning [1], it seems that each year a previously intractable computational task is solved: image recognition, protein folding, human-level text generation…Historically, a major bottleneck when using machine learning to train models was correctly implementing their derivatives in code. One of the key advantages of neural networks is their generic mathematical structure, which has allowed the development of general purpose frameworks that automatically compute derivatives. These frameworks have made it possible to build and train models in a matter of weeks instead of years.

As in other fields, many physicists have exploited the new technology to tackle previously unsolved problems in a wide array of fields. One issue

however with solving physics problems with deep neural networks is that they generally replace the model creation process. The results are models that are simple for machines but inscrutable to humans—they may have excellent predictive accuracy, but lost is the description in terms of a simple effective model. Another common limitation is their inability to generalize beyond the training data: exactly because of their generic structure, neural networks (at least in their most naive form) are completely agnostic to the physical system they should describe. Coupled with their high expressivity, this means that many very different extrapolations beyond the training set are feasible, and thus equally valid.

Regularization methods can help with generalizability; these are generally basic expressions of some "physical" requirement. For example, ridge regression implements a form of minimization of energy, while convolutional architectures encode invariance with respect to translations. Of course, since neural networks are largely inscrutable, determining the correct physical constraints to impose is often more art than science.

Recent developments in "Physics-inspired neural networks" (PINNs) [2] are including increasingly sophisticated physical constraints, but still need to contend with the excess flexibility of neural networks.

In engineering applications, the problem space is usually specified in advance: identify human faces, discriminate malignant tumors, synthesize artificial images, etc. The task of training is to get the network to perform as expected on new data samples of the same type—akin to an interpolation problem, except that the data may span thousands of dimensions.

Science however is all about extrapolation—or, as it is perhaps better referred to in this context, induction. We construct models specifically so that we can predict how systems will behave in new, untested conditions. But how is it that we can have such confidence in a model that we can confidently explain the inner workings of our sun, where the conditions are clearly far removed from anything we can replicate in the laboratory? In essence, we design experiments with the goal to disprove our theory: The more experiments we then perform that don't contradict its predictions, the more it gains our confidence. Induction is thus more about increasing confidence than achieving absolute certainty.

Crucial to this argument is that candidate models are more than just a fit to data, but also a formalization of the scientist's domain expertise. Knowing which features of the data to ignore relies much more on an intuitive understanding of the physics and the intended application regime than on any quantifiable notion of simplicity. In turn, this formalization of domain expertise is why we can expect a theory to remain valid outside of the laboratory. In addition, a theory designed by a human specialist is much more likely to be understandable by other humans, compared to one designed by a machine.

So how does this apply to developing simple theories for complex systems? We discussed above that computational methods methods like deep learning are effective at finding patterns in high-dimensional datasets, but the manner in which they express those patterns is neither intuitive to humans nor appropriate for induction. The answer therefore seems to be that we need to find a way to combine the computational power of machine learning with the domain expertise of practicing scientists. Doing so is the topic of this thesis.

We will do this in two parts, with a focus on the kind of dynamical systems model used in neurophysics. The first part concerns building low dimensional population models, using machine learning to find simplifications that would otherwise be undiscoverable. Of course deriving a low dimensional population

model requires assuming that units are to some extent interchangeable. This is much less true for neurons than say gas particles, and so notable discrepancies remain between the population description and what a more detailed microscopic model would predict. However, by adapting the computational tools used to train neural networks, we can search for effective parameters of the population that better describe the true dynamics.

The workhorse of this method is automatic differentiation. Although developed mostly to train neural networks, this technology can in principle be applied to much more general models. Used in this way, the optimization approach is very generic, and we anticipate that it could be used to find effective parameters for a wide variety of dynamical systems. In Chapter 4 we develop this technology to fit a detailed mesoscale model of neuron populations from recordings of multiple neurons, under the simplifying assumption that the inputs to each population are known.

The second part of this thesis addresses a problem exacerbated by the first. Indeed, if a computer-assisted approach to model discovery is able to search parameter spaces much more intensively, this also tends to generate a large number of plausible candidate models. We are then faced with a new challenge: determining which candidates are "plausible enough" to warrant further analysis. Our answer to the problem is multifaceted, ranging from an ontological reflection to determine what exactly we should be calculating, to developing the mathematical and computational methods to actually perform the desired calculations. This work is explained in Chapter 5.

Finally we close the thesis with a discussion of open problems and possible future directions.

Theoretical background

1. Modelling neurons

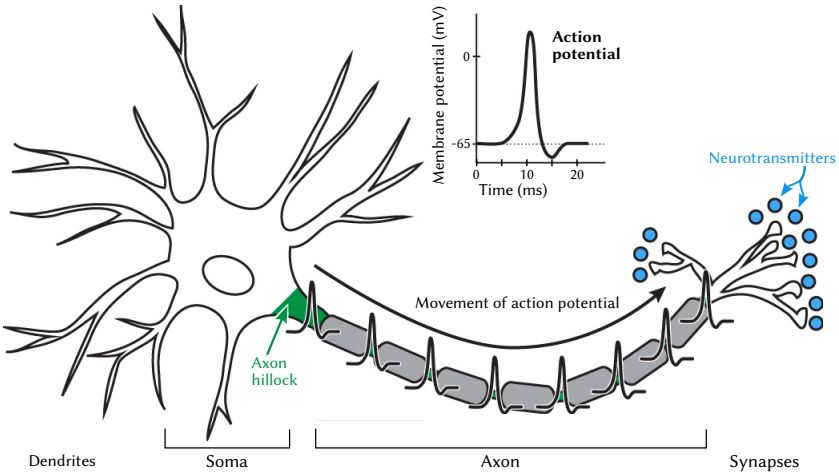
How best to infer a model is entirely dependent on what data we have, and what we know about the system that produced them. In neuroscience the systems of interest range from single neurons, where we might be interested in the precise time course of an electrical potential, to large populations of neurons, where we are more concerned with their aggregate activity. Such different scales of description require very different models. In this first chapter, we describe the types of neuron models that will constitute the motivating examples throughout this thesis, starting with single neurons and working our way up to models for populations composed of tens of thousands of neurons.

1.1. Generating spikes in single neurons

At the population level, the most important feature of neurons is that they generate *action potentials*, or *spikes*. These short electrical pulses, a few milliseconds long, are the primary mechanism by which neurons communicate.

For our purposes, it will suffice to think of a neuron as composed of four main components, schematized in Figure 1.1: the dendrites, the soma, the axon and the synapses. The propagation of information roughly follows the following steps:

1. Action potentials from upstream neurons induce voltage fluctuations in the dendrites, which propagate to the soma.
2. If these are sufficiently strong, the soma initiates a new action potential.
3. The action potential starts at the interface between soma and axon (called the *axon hillock*), and travels down the axon towards the *axon terminal*.



Terminology

Although the terms *action potential* and *spike* can be viewed as synonyms, it is typical to use the former when we care about the *temporal shape* of an action potential, and the latter when we only care about the *time* at which it is initiated.

Figure 1.1.: Neurons interact by sending and receiving voltage fluctuations, known as *action potentials* or *spikes*. At the level of a single neuron, information arrives via *dendrites*, is aggregated in the *soma*, where a "decision" is made about whether to elicit a new spike. If so, it is initiated at the axon hillock and travels along the *axon* until it reaches the *synapses*, which connect to the dendrites of other neurons. Adapted from Henley [3], licensed under CC BY-NC-SA 4.0.

4. The axon terminal contains synapses that release neurotransmitters upon arrival of the action potential. These in turn are picked up by the nearby dendrites of downstream neurons, inducing voltage fluctuations therein and thus continuing the process.

This first section will explain some of the biophysics involved in making this happen, and thereby introduce some of the key quantities that will later appear in our models.

1.1.1. The basic substrate: the cell membrane

Lipids are so-called *amphiphile* molecules: they have a hydro*philic* polar head and hydro*phobic* nonpolar tail. Throw them together in a volume of water, and the hydrophobic tails will tend to clump together to reduce their Gibbs free energy,

forming little balls called micelles (Figure 1.2). These balls can't really grow, since their radius is mostly determined by the length of their tails. Consequently, once the lipid concentration gets so high that micelles start fusing together, they spontaneously change to a less size-constrained shape: a spheroid shell, called a liposome, whose membrane is formed by a *lipid bilayer* (Figure 1.2, bottom).

This type of structure, with a membrane separating an "inside" from an "outside", is one of the key structural elements of a biological cell.[1] Different chemical species will permeate more or less easily through a membrane, and cells can modulate that permeability — on a species-specific basic—with passive channels and active pumps. All kinds of interesting biophysics can be produced this way; for our purposes, we are most interested in the modulation of three species — sodium (Na^+), potassium (K^+) and calcium (Ca^{2+}) — which are involved in the generation of fast action potentials that ultimately support cognition.

These action potentials are very fast, on the order of a few milliseconds—too fast to be produced by ion pumps. Instead, pumps are part of the background homeostatic mechanism, expending energy to move ions across the membrane and maintain a concentration gradient. Then, at the appropriate moment, a neuron will open channels and let ions rush in or out, producing the characteristic voltage spike.

1.1.2. From concentration gradient to electrical Nernst potential

An important consequence of the cell maintaining a concentration gradient is that this also produces a difference in *electric* potential between the intra- and extra-cellular sides of the membrane; typically the inner side is about 70 mV lower. Note that a lipid bilayer has the thickness of just two molecules, meaning that this voltage is sustained over a very small distance (about 3 nm)—indeed, these structures have quite large capacitances of around $1\mu F/cm^2$ [4, 5].

To quantify the relationship between ion concentration and membrane potential, we consider small intra/extra-cellular volumes V^{in} and V^{out} on either side of the membrane and approximate the dissolved ions as a ideal gas so

1: The term "liposome" is usually reserved for the nanoscale lipid-bilayer containers that assemble spontaneously. Cells are orders of magnitude bigger — comfortably in the microscale — and are supported by cytoskeletons. Their outer membrane is still formed from a lipid-bilayer, but it now houses an array of proteins that allow the cell to interact with its environment.

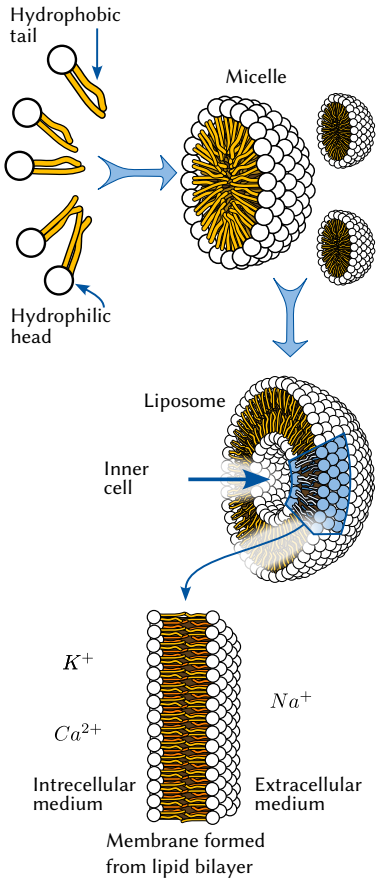


Figure 1.2.: Spontaneous formation of a lipid bilayer. Cells use their membrane to create *concentration gradients* between the inside and outside of the cell. Differences in concentration between Na^+, K^+ and Ca^{2+} ions, maintained with ion pumps, are what drive neural activity. (Image by Mariana Ruiz Villarreal; public domain. https://commons.wikimedia.org/wiki/File:Phospholipids_aqueous_solution_structures.svg)

that we can treat each species independently.[2] We have N_a^{in} ions in V^{in} and N_a^{out} ions in V^{out}; ions can pass through the membrane but their total number $N_a^{\text{in}} + N_a^{\text{out}}$ is constant,[3] so we can write

$$N_a^{\text{in}} =: N \qquad N_a^{\text{out}} = -N_a \,. \tag{1.1}$$

This is analogous therefore to a simple chemical reaction where one species (ions inside the cell) is exchanged for another (ions outside the cell). Schematically:

$$a^{\text{in}} \rightleftharpoons a^{\text{out}}. \tag{1.2}$$

We will define the extracellular medium has having zero potential, so the that membrane potential ΔV is equal to the potential inside the cell:

$$E^{\text{out}} \equiv 0 \qquad E^{\text{in}} = \Delta V \,. \tag{1.3}$$

Chemical reactions seek to minimize their Gibbs energy G.[4] Energy is intensive, so the total energy is the sum of that on both sides of the membrane:

$$G = G^{\text{in}} + G^{\text{out}} = U_a^{\text{in}} + U_a^{\text{out}} + PV^{\text{in}} + PV^{\text{out}} - TS_a^{\text{in}} - TS_a^{\text{out}} \tag{1.4}$$

Since transport across the membrane equates to changing the number N_a of ions inside the cell, chemical equilibrium of Equation (1.2) is equivalent to the condition

$$\frac{\partial G}{\partial N_a} = \frac{\partial (U_a^{\text{in}} + U_a^{\text{out}})}{\partial N_a} - T \frac{\partial (S_a^{\text{in}} + S_a^{\text{out}})}{\partial N_a} \overset{!}{=} 0 \,, \tag{1.5}$$

so we need to express U_a and S_a as functions of N_a. (The other state variables P, V and T are independent of N_a.)

For the internal energy U, the only contribution that matters is that due to the potential difference (any other contributions, e.g. from thermal fluctuations, are the same on both sides and thus independent of N_a), so we write:

$$\begin{aligned} U_a^{\text{in}} &= N_a^{\text{in}} z_a e E^{\text{in}} + \text{cst} = N_a z_a e \Delta V + \text{cst} \\ U_a^{\text{out}} &= N_a^{\text{out}} z_a e E^{\text{out}} + \text{cst} = \text{cst} \,, \end{aligned} \tag{1.6}$$

where e is the elementary charge and z_a is the charge of each ion in units of e (so $z_{\text{Na}^+} = 1$ while $z_{\text{Ca}^{2+}} = 2$). Which leaves us with

$$\frac{\partial (U_a^{\text{in}} + U_a^{\text{out}})}{\partial N_a} = z_a e \Delta V \,. \tag{1.7}$$

For the entropy S, we can start from the fundamental relation $S = k \log \Omega$, where k is the Boltzmann constant and Ω the total number of possible configurations. Since we assumed an ideal gas, we know Ω will be proportional to the number of ways we can arrange ions within a volume V, i.e.

$$\Omega_a^{\text{in/out}} \propto \frac{(V^{\text{in/out}})^{N_a^{\text{in/out}}}}{N_a^{\text{in/out}}!} \,, \tag{1.8}$$

where the denominator accounts for the non-identifiability of ions. Applying Stirling's approximation, we have

$$S_a^{\text{in/out}} = k \log \Omega^{\text{in/out}} \approx k N_a^{\text{in/out}} \left[1 - \log \frac{N_a^{\text{in/out}}}{V^{\text{in/out}}} \right], \tag{1.9}$$

2: This presentation is based in part on the treatment of chemical reactions in §6.6 of [6], simplified on the one hand to consider only one reactant (N_a^{in}) and one product (N_a^{out}), but extended to consider separate volumes.

3: Of course the other surfaces of V^{in} and V^{out} are actually open, but since they are at equilibrium with their surroundings, as many ions exit through them as come in.

4: This is because reactions occurring in solutions are usually at constant pressure and temperature. Since pressure (P) and temperature (T) are independent variables of the Gibbs energy (one may recall that its differential form is $dG = dU + VdP - SdT + \sum_a \mu_a N_a$), we simply leave them constant for the minimization.

In Equation (1.4), U, P, V, T and S are respectively the internal energy, pressure, volume, temperature and entropy.

which we can differentiate to obtain[5]

$$\frac{\partial(S_a^{\text{in}} + S_a^{\text{out}})}{\partial N_a} = -k \log \frac{N_a^{\text{in}}}{V^{\text{in}}} + k \log \frac{N_a^{\text{out}}}{V^{\text{out}}} \,. \tag{1.10}$$

We can rewrite Equation (1.10) in terms of concentrations on either side of the membrane, where $[a]$ is the concentration of ion a (conventionally in moles per litre, although here the units don't matter):

$$\frac{\partial(S_a^{\text{in}} + S_a^{\text{out}})}{\partial N_a} = -k \log \frac{[a]_{\text{in}}}{[a]_{\text{out}}} \,. \tag{1.11}$$

Inserting Equation (1.7) and Equation (1.11) into Equation (1.5), we get the condition

$$kT \log \frac{[a]_{\text{in}}}{[a]_{\text{out}}} \stackrel{!}{=} z_a e \Delta V_a \,, \tag{1.12}$$

known as the *Nernst equation* [7, §2.1.1, 8, §2.1.1, 9], for zero net diffusion through the membrane.

To understand the implication of Equation (1.12), it is important to remember that the species concentrations $[a^{\text{in}}]$ and $[a]_{\text{out}}$ are actively maintained by the cell's ion pumps. For most ion species they can be considered constant, so that the potential E_a that satisfies Equation (1.12)—called the *Nernst potential*—is simply a parameter of the model.[6]

1.1.3. Ion flows through a membrane

The Nernst potential E_a solving Equation (1.12) can be understood as an equilibrium between two forces: one *electric* (Equation (1.7)), the other *entropic* (Equation (1.11)). To generate an action potential however, we can't always have the membrane potential V be equal to E_a: this sets up a net *current* of a ions through the membrane. To first order around the equilibrium, this current can be approximated as a linear function of the difference between V and E_a:

$$I_a = \bar{g}_a(\Delta V - E_a). \tag{1.13}$$

In other words, in this linear regime, the ultimate effect of the ion pumps maintaining a concentration gradient across the membrane is to produce the equivalent of an ohmic voltage source with potential E_a. We interpret the proportionality constant $\bar{g}$ as the *conductance* of the membrane.

Most ion species involved in producing action potentials have positive charge, so that they flow in the direction of the electric field:

Membrane potential	Ion flow direction	Current
$\Delta V = E_a$	None	$I_a = 0$
$\Delta V < E_a$	*Into* the cell	$I_a < 0$
$\Delta V > E_a$	*Out* of the cell	$I_a > 0$

Since it is the potential at which the ion current changes direction, E_a is also called the *reversal potential* for ion species a.

5: Recall that $\frac{\partial}{\partial N_a} = \frac{\partial}{\partial N_a^{\text{in}}} = -\frac{\partial}{\partial N_a^{\text{out}}}$.

6: An exception is the intracellular concentration of Ca^{2+} ions, which can vary sufficiently that it needs to be accounted for in equations. We will see an example of this in Section 1.2 (p. 15). Even then however, *extra*cellular concentrations of Ca^{2+} remain effectively constant, due to both larger outside volume and concentration.

Units

Note that since ion flow is distributed over the membrane, the units of I_a in Equation (1.13) are ampere *per area*.

Important

While the flow of ions across the membrane does not meaningfully change the concentration gradient, it *is* enough to change the membrane potential ΔV.

The ion current described by Equation (1.13) mostly takes place through *ion channels*.[7] These are complex folded proteins embedded into the membrane; many are ion-selective, i.e. they are only conductive for a specific ion species. There are also many different channels for each ion species; the potassium channel illustrated in Figure 1.3 is one of the simpler varieties.

The conductance $\bar{g}_a$ in Equation (1.13) (which we recall is actually conductance *per area*) is therefore a function of these channels: more a channels per μm makes the membrane more permeable to a ions, and therefore more conductive for that ion species. For modelling action potentials, the important channels are those that are *voltage-gated*: these can be either opened or closed, with a probability that depends on the membrane potential. The conductance of the membrane depends on the number of open channels, and thus ultimately on its membrane potential.

If we look at a small patch of membrane, illustrated in Figure 1.4, we effectively have two parallel plates (formed by the hydrophile heads of the lipid bilayer) and a potential source (modelled by Equation (1.13)) maintaining a potential difference between those plates. In other words, the membrane behaves very much like a resistor-capacitor (RC) circuit, except that instead of an external wire connecting the two plates, the flow of ions occurs over their entire surface through a multitude of channels. From the point of view of modelling the total voltage across the membrane, we can reduce the entire system to the equivalent circuit illustrated in Figure 1.4b.

If there is only one ionic species present, how the membrane potential changes over time is therefore also modelled by the usual equations for an RC circuit,[8]

$$C\frac{dV}{dt} = I_a = \bar{g}_a(V - E_a)\,. \tag{1.14}$$

where C is the capacitance *per area* of the membrane.

1.1.4. More ion species: the conductance model

The derivation of the Nernst equation (1.12) assumes that only a single ion species flows through the membrane — after all, the Nernst potential E_a is different for each species, but there can only be one potential difference across the membrane. One can extend the above argument to the case with multiple ion species and thus obtain the Goldman-Hodgkin-Katz formula for ion fluxes [9]; however this is rarely done in practice [7, §2.1.2, 8, §2.2].

Instead we usually stay within the first-order approximation, determining the flow of each ion with Equation (1.13). Since the different ions flow independently, each through dedicated channels,[9] the total ionic current is simply the

7: Some diffusion does happen through the membrane, especially for K^+ ions. However, as the rate does not depend on the membrane potential, this can be treated as part of the background processes that work together to maintain the concentration gradient.

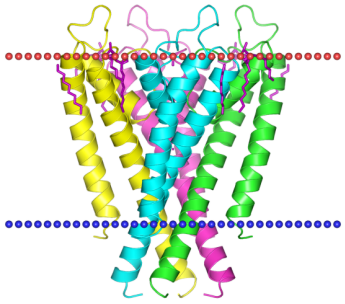
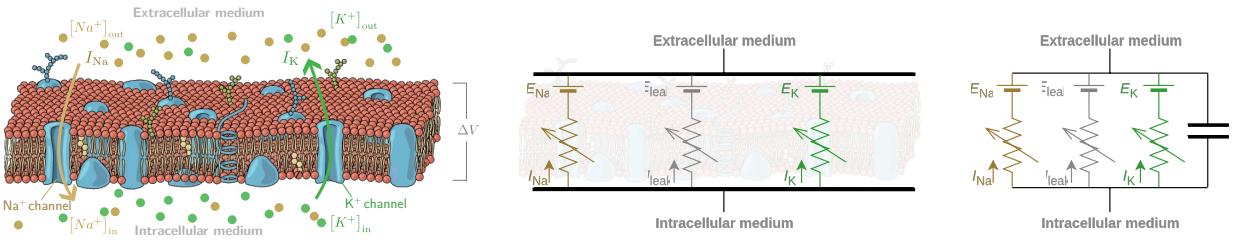


Figure 1.3.: Potassium channel KcsA: colours indicate the four subunits, and dotted lines indicate the boundaries of the lipid bilayer membrane. Configuration calculated for the *Orientations of Proteins in Realistic Lipid Membranes database* [10]; image retrieved from https://opm.phar.umich.edu/proteins/59 on 25 August 2025.

8: From here on we write V (instead of ΔV) for the membrane potential, since there is no risk of confusion with a volume variable.

9: We should note here that there are in fact many different types of channels for each ion, each with different kinetics for the conductance g. Although we write Equation (1.14) in terms of a sum over ion species to keep the exposition linear, in practice the sum would actually be over channel types. The parallel circuit equivalence is a consequence of the independence of different channels, so whether or not those channels carry different ion species is not really important.



(a) **(b)**

Figure 1.4.: The lipid bilayer as an RC circuit The conductance model replaces a patch of neuron membrane with an equivalent RC circuit with parallel voltage sources. Each voltage source corresponds to a different *ion channel type*. In contrast to the electrical analog, here the model describes a *density* of current, since channels are distributed across the entire surface of a neuron. **(a)** A lipid bilayer is suffused with many different proteins, among them ion channels that are permeable to specific ion species. Channels can be open or closed; when open, the net ion current through them (here illustrated as I_{Na}) depends on the balance between entropic forces (stemming from the concentration gradient) and electric forces (due to the potential gradient). To first order, the flow is linear in the potential.
Adapted from Betts et al. [11], licensed under CC BY 3.0. **(b) (left)** The dynamics are well-approximated by an RC circuit: voltage-gated ion channels are akin to variable resistances, and the membrane itself acts like a capacitor. The current through a channel for ion a is ohmic when the voltage is measured with respect to that ion's Nernst potential E_a: this is equivalent to giving each channel a potential source fixed to $-E_a$ (Equation (1.13) (p. 9)). An additional *leak* current, not associated to any particular channel, is usually included to account for unmodelled diffusion through the membrane. Note that in this analogy, the pumps and channels are inside the plates of the capacitor. **(right)** Equivalent RC circuit drawn in a more conventional layout.

sum of the currents, so that Equation (1.14) becomes

$$\begin{aligned} C\frac{dV}{dt} &= I_{K^+} + I_{Na^+} + I_{Ca^{2+}} + I_{leak} \\ &= \bar{g}_{K^+}(V - E_{K^+}) + \bar{g}_{Na^+}(V - E_{Na^+}) \\ &\quad + \bar{g}_{Ca^{2+}}(V - E_{Ca^{2+}}) + \bar{g}_{leak}(V - E_{leak})\,. \end{aligned}$$

The "leak" current is included to account for additional channels that are not explicitly modelled. This is of course analogous to the RC circuit in Figure 1.4b, where each ion current is a different voltage connected to the capacitor in parallel.

1.1.4.1. Dynamic conductances lead to dynamic neurons

As we alluded to above (p. 9), the important channels for generating action potentials are those whose conductance g depends on the membrane potential. For our purposes, it suffices to consider channels as composed of four independent subunits (see Figure 1.3), each of which can be in an "open" or "closed" state — all subunits must be "open" for ions to be allowed through a channel.

Note

Ion currents are normally indexed by the channel type, as there can be multiple channel types for each ion species. Below we use i to index channels, to distinguish from the a we used above for ions.

The conductance through a membrane patch for a particular channel type is then determined by the fraction of channels of that type that are currently open. If $m_i(t)$ is the probability of a single subunit to be "open" at time t, and g_i is the maximum conductance assuming all open channels, this means we

can write the current through channels of type i as

$$I_i = g_i m_i(t)^4 \,. \tag{1.15}$$

In this form, m is called the *activation variable*. In many channels, one of the subunits differs from the others: it tends to close when the others are open.[10] These are conventionally represented by an inactivation variable h:

$$I_i = g_i m_i(t)^3 h_i(t) \,. \tag{1.16}$$

A key feature of activation variables is that they don't react instantaneously to changes in the membrane potential V. There is a voltage-dependent stationary value $m_\infty(V)$ ($h_\infty(V)$) to which they converge, but they do so on a time scale τ_m (τ_h):

$$\tau_m^i \frac{dm_i}{dt} = m_\infty^i(V) - m_i \qquad \tau_h^i \frac{dh_i}{dt} = h_\infty^i(V) - h_i \,. \tag{1.17}$$

The values of τ_m^i and τ_h^i (which in general also depend on V), as well as the functions m_∞^i and h_∞^i, characterize each channel and must be determined experimentally. For examples values and functions, see Section 1.2 (p. 15) below. Note in particular how shapes of the h_∞ functions are inverted compared to m_∞.

Equations. (1.15) to (1.17), along with the generalized form of Equation (1.18),

$$C\frac{dV}{dt} = \sum_{i \in \text{channels}} g_i \, m_i^p(t) \, h_i(t) \, (V - E_i) \,, \tag{1.18}$$

define what is referred to as the *conductance model* [7, 8, 12, 13].[11] Although still highly simplified compared to the real neurons, this model is usually sufficient to model the dynamics of the membrane potential [12, p. 11].[12] The dependence of activation variables on the membrane potential V sets up a feedback loop whereby changes V produce nonlinear changes in the channel conductances g_i — each on with their own voltage-dependent time scales τ_m^i and τ_h^i — which in turn change the ion currents and thus the membrane potential. These interactions provide us with sufficiently rich dynamics to model the entire course of an action potential.

Typical phases of an action potential

Figure 1.5 depicts an action potential, along with the corresponding time courses of the activation variables for the main Na^+ and K^+ currents. We can identify three main phases:

1. The membrane *depolarizes* (i.e. V become less negative), causing Na^+ channels to activate. More Na^+ ions enter the cell, further depolarizing the membrane.
2. If the depolarization crosses a threshold, this becomes a runaway process, completely depolarizing the cell in ~1 ms.
3. Sodium channels become inactivated. At the same time, potassium channels activate, causing K^+ ions to exit the cell and restoring the membrane potential in ~2–4 ms.

After the membrane's resting potential has been restored, and over a much longer time scale, ion pumps exchange Na^+ and K^+ ions to restore ion concentrations to their initial levels.

10: This is the mechanism by which the flow of Na^+ ions is shut off after the initial spike of an action potential.

The relation to the equations above is obtained with either $\bar{g}_i(t) = g_i m_i(t)^4$ or $\bar{g}_i(t) = g_i m_i(t)^3 h_i(t)$.

In Equation (1.18), the parameter p is either 3 or 4, depending on whether channel i is modelled with ($p = 3$) or without ($p = 4$) an inactivation variable. In the latter case, $h_i(t)$ is formally set to a constant function $h_i(t) \equiv 1$.

11: The first model of this form was proposed by Hodgkin and Huxley [14], which is why conductance models are also sometimes called "Hodkgin-Huxley-like models".

12: One exception is the contribution of the calcium current, for which the intracellular concentration $[Ca]^{in}$ changes too much for the Nernst potential to be considered constant. Even then, this can be accounted for by adding $[Ca]_{in}$ as a dynamic variable, and then updating the Ca^{2+} Nernst potential with the help of Equation (1.12) (p. 9) (see Equation (1.22) (p. 17) and [15]).

Important

Since phases 2 and 3 are mostly driven by processes internal to the neuron, once an action potential is initiated, the rest of the V dynamics are largely stereotyped.

1.1.5. Features of action potentials: refractory periods and adaptation

The conductance model presented in Section 1.1.4.1 (p. 11) is intended to model the membrane potential at the cell's soma, where spikes are initiated (recall Figure 1.1 (p. 6)). Although dynamically rich, one should keep in mind that this model is already highly simplified. For instance, we have not considered

- that neurons have spatial extent, and that action potentials must propagate along axons;
- that the density and type of ion channels are not spatially homogeneous, and in particular differ in the dendrites, soma and axon;
- that somewhat different action potentials are also initiated in dendrites;
- that these dendritic action potentials interact before even reaching the soma;
- that synapses have their own dynamics, tied to depletion of neurotransmitters;
- that the ion pumps require time to restore ion concentrations;
- that the opening/closing of channels is a thermodynamic process, and therefore stochastic.

These features are essential to understanding how single neurons function. However at the scale of large populations of neurons, we usually absorb them into network parameters like the communication delay or connection strength between neurons. Some features however are important enough to merit explicit treatment at the population level; we highlight two that will be important in later chapters:

Refractory period During the few milliseconds of an action potential, a neuron more or less completely depolarizes to around 0 mV, from its resting potential of around -65 mV. In contrast, during subthreshold activity inputs from other neurons typically cause fluctuations on the order of only a few millivolts over a similar time frames. As a consequence, once an action potential is initiated, a neuron is effectively impervious to further input.

We call this time window during which a neuron is unresponsive the *refractory period*.

Adaptation If a neuron fires repeatedly, especially during a *burst* of spikes, we observe experimentally that it seems to "tire": it requires increasingly stronger depolarizing inputs to trigger new spikes. Phenomenologically, this is usually described in one of two ways: either by decreasing the membrane potential after a spike, or by increasing the *firing threshold* it must cross in order initiate a new spike.

Biophysically, one of the main reasons for adaptation are changes in ion concentration [8, §6.3.2]. With each spike, a number of ions travel across the membrane; although we previously assumed the ion concentrations inside and outside the cell to remain constant, this of course cannot be exactly true. The small changes in concentrations lead to changes in the ionic currents that

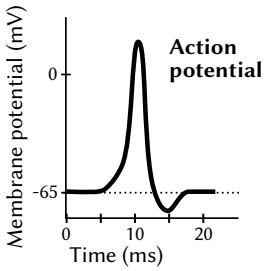


Figure 1.5.: Typical time course of an action potential. Initially the Na^+ conductance is high, causing the membrane to depolarize. A K^+ current then sets in in the reverse direction, repolarizing the membrane potential. (Inset from Figure 1.1 (p. 6).)

For more on modelling the biophysical details of single neurons, see for example [9] or [16]. The University of Ottawa is also active in this area, including work by my colleagues Barlow [17], Jacques [18] and Lee [19].

are unfavourable to further spiking, changes which persist for some time even after the membrane has returned to its resting potential, until the combination of passive (via ion current) and active (via ion pumps) transport has returned ion concentrations to their resting state.

This period during which firing is suppressed but not impossible is sometimes called the *relative refractory period* [20, 8, §2.3.3].

1.1.6. Wiring neurons together: electrical and chemical synapses

For all of the fascinating biophysics at the single-neuron level, computation mostly happens at the level of *networks* of neurons. For this of course we need to connect neurons somehow, so that the spikes triggered in one upstream neuron can affect the membrane potential of other downstream neurons.

As we mentioned at the start (p. 6), communication between neurons occurs via synapses, of which there are two main types: *chemical* and *electrical*.

Terminology

In neuroscience, "upstream" and "downstream" are more commonly referred to as *presynaptic* and *postsynaptic*. One advantage of localizing the terminology to the synapse is that it does not presume a global direction to spike propagation.

1.1.6.1. Electrical synapses

An electrical synapse is akin to a direct, narrow connection between the terminals of the presynaptic axon and postsynaptic dendrite, allowing ions to flow directly from one to the other. (See Figure 1.6.)

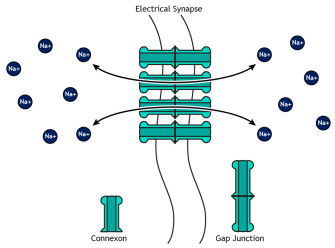


Figure 1.6.: An electrical synapse allows ions to flow between the neurons, functioning like a resistor in a circuit.
Reproduced from Henley [3], licensed under CC BY-NC-SA 4.0.

Ion flow through the synapse is nevertheless more restricted than within a cell, so that mechanistically the synapse acts like a resistor: the current through the synapse depends on the membrane potentials on either side and is given by Ohm's law. If we write V_{pre} for the potential at the axon terminal (called the *pre*synaptic potential) and V_{post} for the potential at the dendrite terminal (the *post*synaptic potential), the current I_e through an electrical synapse is given by

$$I_e = g_e(V_{\text{post}} - V_{\text{pre}}), \tag{1.19}$$

where g_e is the conductance of the synapse. Although certainly important, electrical synapses are far less common than chemical synapses, and often neglected in models—either omitted entirely, or given a conductance parameter clearly selected based on convenience rather than experimental evidence.[13]

13: For example, to the best that I could ascertain, Marder and Abbott [12] and Prinz, Billimoria, and Marder [15] simply set $g_e = 1$.

1.1.6.2. Chemical synapses

In chemical synapses, there is a gap—called the *synaptic cleft*—between the presynaptic and postsynaptic neurons, so ions cannot flow directly from one to the other (see Figure 1.7). Instead, the axon terminals contain special compounds called *neurotransmitters*,[14] which are released into the cleft when an action potential arrives. These diffuse across the cleft and bind to the postsynaptic dendrites, causing them to increase conductance of their membrane. If their numbers are sufficient, the conductance becomes large enough to cause a runaway process, and a dendritic postsynaptic action potential is initiated.

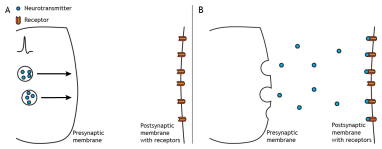


Figure 1.7.: In a chemical synapse, neurotransmitters modulate the conductance of the postsynaptic neuron.
Reproduced from Henley [3], licensed under CC BY-NC-SA 4.0.

In this sense, the initiation of a dendritic postsynaptic action potential mirrors that of the action potential in the soma [13, §5.8], with neurotransmitters playing a gating role. One notable difference is that we now have two separate

14: Some common neurotransmitters are *glutamate*, *GABA* and *acetylcholine*. Typically action potentials are facilitated by glutamate and inhibited by GABA and acetylcholine, but this is not always the case.

membrane potentials: the *pre*synaptic potential controls the activation (i.e. gating), while the *post*synaptic potential determines the current flow.

For an example of how these synapses are modelled mathematically, we consider the model of Prinz et al.:[15]

15: [21]. See also [12].

$$I_s = g_s s(V_{\text{post}} - E_s) \tag{1.20a}$$

$$\tau_s \frac{ds}{dt} = s_\infty(V_{\text{pre}}) - s \tag{1.20b}$$

$$s_\infty(V_{\text{pre}}) = \frac{1}{1 + \exp\big((V_{\text{th}} - V_{\text{pre}})/\Delta\big)} \tag{1.20c}$$

$$\tau_s = \frac{1 - s_\infty(V_{\text{pre}})}{k_-} \tag{1.20d}$$

Here s is the synaptic activation variable; it has a voltage-dependent stationary value $s_\infty(V_{\text{pre}})$, but as with ion channels, changes in s are not instantaneous. The equations are parameterized by a maximum conductance g_s, a postsynaptic reversal potential E_s, a voltage sensitivity Δ and a rate constant k_-.

To understand how Equation (1.20) leads to a postsynaptic action potential, recall that when an action potential arrives, the cell membrane briefly depolarizes, i.e. V_{pre} becomes large. We see from Equation (1.20c) that a larger V_{pre} entails a larger s_∞ (up to $s_\infty = 1$ when $V_{\text{pre}} \gg V_{\text{th}}$). The activation variable s then approaches s_∞ according to Equation (1.20b), increasing the postsynaptic current as per Equation (1.20a), in close analogy to the conductance model (p. 11).

In contrast to the conductance model, the sharp spike in postsynaptic current comes not from a runaway positive feedback, but is more-or-less directly baked in with Equation (1.20d): as s_∞ approaches one, the time scale τ_s decreases, leading to faster changes in s and therefore I_s. This simpler model is a reflection not only of the different physiology of synapses, but also that when it comes to modelling circuits with multiple cells, the activity at the somae tends to be the most important.

16: "*pyloric*: relating to or affecting the region where the stomach opens into the duodenum." (*Oxford Dictionaries*, retrieved August 22 2025.)

Notation

The conductance model is described in Prinz, Billimoria, and Marder [15] (p.1-2, §*Model*). Where differences in notation occur, we prefer that in Prinz, Bucher, and Marder [21].

1.2. Example of a simple circuit: the three component model for the crustacean pyloric rhythm

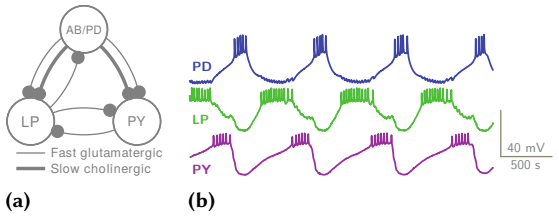
One way to study how emergent behaviour arises from the interactions between neurons is to consider small circuits. As an example, we here introduce the three component model of crustacean stomatogastric ganglion [12, 15, 21], used to explain the triphasic firing pattern—referred to as the *pyloric*[16] *rhythm*—observed in that system. Variants of this seminal model continue to be used to this day, in particular to study compensatory mechanisms [22], and we will use it as a case study in Chapter 5 (p. 99).

Prinz, Bucher, and Marder [21] model the lobster's pyloric circuit using 9–14 neurons grouped into three populations. As shown in Table 1.1, populations contain neurons of one or two cell types—one of AB, PD, LP or PY—and are connected via a dense set of inhibitory synapses: only PY neurons don't directly inhibit AB/PD neurons (Figure 1.8a). Neurons are modelled with the previously described conductance model (p. 11), and connected with a

One may wonder how a fully inhibitory circuit can produce autonomous activity. The reason is at least partly that these neurons exhibit a phenomenon called *post-inhibitory rebound* [23, 8, §2.3.5], where sustained inhibition can cause the neuron to fire when that inhibition is released. Post-inhibitory rebound typically requires at least two dynamic variables to model properly, and is part of the repertoire of neuron behaviours that we neglect when we reduce neurons to 1-D accumulators, as we do in Section 1.4 (p. 20).

Population name	Cell type names	Number of cells
AB/PD	anterior bursting	AB: 1
“pacemaker kernel”	pyloric dilator	PD: 2
LP	lateral pyloric	1
PY	pyloric	5–8

Table 1.1.: Neuron populations of the pyloric network



(a) **(b)**

Figure 1.8.: Triphasic rhythm in the pyloric circuit (a) Circuit diagram of the pyloric network, reproduced from Fig. 1 of Prinz, Bucher, and Marder [21]. All chemical synapses are inhibitory. **(b) Electrophysiological recordings** from analogue neurons in the crab, exhibiting characteristic triphasic activity. Reproduced from Kispersky, Gutierrez, and Marder [23].

mixture of chemical and electrical synapses; the spatial extent of neurons, and in particular the time it takes action potentials to propagate down an axon, is not considered.

As mentioned, a defining feature of this model is its triphasic firing pattern. The pattern is driven by the cells AB and PD, which are qualified as “pacemakers” due to their strong autonomous[17] firing pattern that regularly alternates between active and quiescent states (see the PD traces in Figure 1.8b). These cells drive the circuit. The other cell types modulate the circuit’s activity, firing during the other cells’ quiescent phases and thus establishing the three activity phases shown in Figure 1.8b.

17: *Autonomous* here means that the neurons initiate and sustain the observed firing pattern without any external input. This is in contrast to other fully inhibitory circuits that need constant input in order to remain active.

The modelling framework of Prinz, Bucher, and Marder [21] distinguishes neuron *types* from neuron *models*:

Neuron types are one of AB, PD, LP or PY. In the model, a cell’s type determines the activation kinetics of its ion channels, as well as the type of synapses it makes with other neurons. These were determined through experiments and are always the same. The only parameters *not* specified by the cell type are the maximum conductances $\bar{g}$ in Equation (1.18) (p. 12) and the strength of the synapses (parameter g_s in Equation (1.20a)).

Neuron models are variants of each cell type with different maximum conductances for each of their ion channels. Each model corresponds to one cell type and a vector of 8 maximum conductance values $\bar{g}$, which were determined by sweeping the parameter space to identify combinations that produce activity similar to that observed in experiments for each cell type [15].

Each realization of the circuit is defined by specifying one neuron model for each population. AB and PD neurons are always given the same model, which is why they can be modelled as one population; the only difference between them is that the AB cell has glutamatergic synapses while the PD cells have slower cholinergic synapses.

	I_{Na}	I_{CaT}	I_{CaS}	I_{A}	$I_{\text{K(Ca)}}$	I_{Kd}	I_{H}	I_{leak}	unit
E_i	50	E_{Ca}	E_{Ca}	-80	-80	-80	-20	-50	mV
p	3	3	3	3	4	4	1		

Table 1.2.: Ion channel parameters for the dynamics defined in Equation (1.21). Values are taken from Prinz, Billimoria, and Marder [15].

1.2.1. Single compartment conductance model

Each cell type is modelled with the conductance model (c.f. Equation (1.14) (p. 11)),

$$\begin{aligned} \frac{C}{A}\frac{dV}{dt} &= -\sum_i I_i - I_{\text{input}} \\ I_i &= g_i m_i^p h_i (V - E_i) \\ I_{\text{input}} &= I_e + I_s + I_{\text{ext}}\,, \end{aligned} \tag{1.21}$$

with currents through six selective ion channels: one sodium channel (I_{Na}), two calcium channels (I_{CaT}, I_{CaS}), three potassium channels (I_{A}, $I_{\text{K(Ca)}}$ and I_{Kd}). Two additional currents are also modelled: a "hyperpolarization-activated" channel is permeable to both Na^+ and K^+ and produces the current I_H, and a catch-all leak current I_{leak} models any remaining currents. The values of p and E_i for each current are listed in Table 1.2. Note also that three channel types do not have inactivating variables (i.e. they are not present in $I_{\text{K(Ca)}}$, I_{Kd} and I_{H}); the corresponding h_i can be formally set to 1 for consistency of notation.

In Equation (1.21), I_i is the current for each channel, while I_{input} is the current from synaptic inputs I_e and I_s. These are computed using the electrical (p. 14) and chemical (p. 14) synapse models defined above. We also allow for an additional external input current I_{ext}; this current is not necessary to drive the system (after all, a defining feature of the pyloric circuit is its spontaneous rhythm) but can be used to model inputs from other neuron populations, or currents injected by the experimenter. Expected magnitudes for I_{ext} are 3–6 nA.[18]

18: Strictly speaking, the units of I are actually nA/cm^2; reporting I in units of nA implies the convention $C/A = 1$. A more correct, although less conventional, reporting of this magnitude would be $I_{\text{ext}}\frac{A}{C} = 3\text{–}6\text{mV/ms}$.

The key variables here are g_i, m_i and h_i, which we already described in Section 1.1.4.1 (p. 11). The first is the *maximum conductance* of channel i; its value is determined by the *neuron model*. Maximum conductance values for the models identified by Prinz, Billimoria, and Marder [15] are given in Table A.4 (p. 157). The variables m_i and h_i are the channel's activation and inactivation variables, whose kinetics are described by Equation (1.17) (p. 12). Their stationary functions m_∞ and h_∞ were determined experimentally; a few of the resulting fitted curves are shown in Figure 1.9; for the full set, see Figure A.2 (p. 159).

Calcium concentrations in this system change sufficiently to affect the reversal potential of calcium currents: extracellular concentrations can be assumed fixed to 3 mM, but intracellular concentrations are dynamic and evolve according to

$$\tau_{\text{Ca}}\frac{d\left[\text{Ca}^{2+}\right]_{\text{in}}}{dt} = -f(I_{\text{CaT}} + I_{\text{CaS}}) - \left[\text{Ca}^{2+}\right]_{\text{in}} + \left[\text{Ca}^{2+}\right]_0\,, \tag{1.22}$$

where f is a scalar parameter. Knowing the Ca^{2+} concentrations, we can then update the reversal potential by solving the Nernst equation (Equation (1.12)

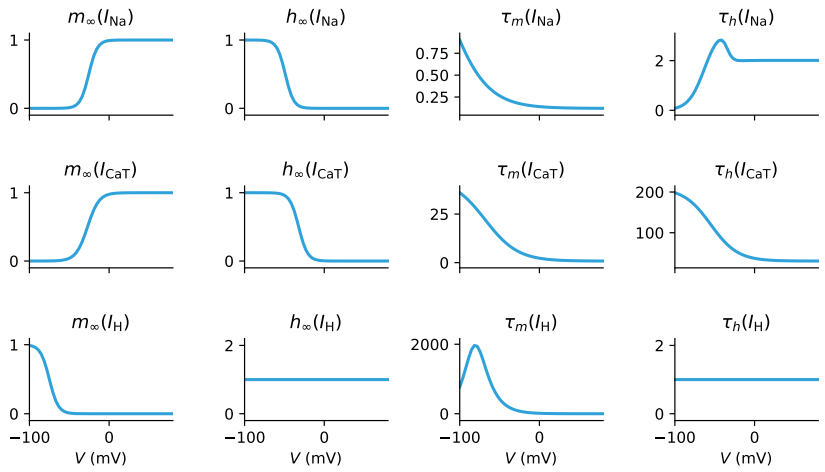


Figure 1.9.: Subset of voltage-dependent stationary values for the pyloric circuit. Note that m_∞ increase with V, while h_∞ decrease, consistent with their respective interpretation as *activation* and *inactivation* variables. No h_∞ and τ_h curves are shown for I_H since it does not have inactivating variables. For the full set of curves, see Figure A.2 (p. 159).

(p. 9) from Section 1.1.2 (p. 7)):

$$E_{Ca} = \underbrace{\frac{kT}{ze}}_{=:\gamma} \ln \frac{[Ca^{2+}]_{out}}{[Ca^{2+}]_{in}} = \gamma \ln \frac{[Ca^{2+}]_{out}}{[Ca^{2+}]_{in}} \quad [mV]\,. \tag{1.23}$$

For full details on our implementation of this model, including all parameter values, see Chapter A (p. 153).

1.3. Apparté: Balancing complexity and identifiability of models

Often when modelling a system, our instinct is to include everything we know to make the model as realistic as possible. However, it is often the case that more realistic models lead to *worse* predictions: more parameters also means more interaction mechanisms, and therefore more ways for things to go wrong.

A model is said to be *identifiable* if all of its parameters can be uniquely determined given unlimited amounts of data. We can illustrate this with a linear model that learns a matrix A to predict $y \in \mathbb{R}^2$ from $x \in \mathbb{R}^3$,

$$y = Ax\,. \tag{1.24}$$

In Equation (1.24), all of the coefficients of A are identifiable as long as the x are sampled from a non-degenerate distribution.[19]

We can make the model *non*-identifiable by making it *overparameterized*, for example with an additional unknown matrix B:

$$y = Ax + Bx\,. \tag{1.25}$$

Now it is impossible to identify both A and B: since $y = (A + B)x$, we can only ever solve for $A + B$. (An overparametrized model has *symmetries*[20] under which it is invariant.) Note that this can occur even if the dependent variable y in Equation (1.25) really is the result of adding two linear effects:

19: An easy to see this is that if the x are sampled from all of $\mathbb{R}^3$, we will eventually obtain the samples $(1,0,0)^T$, $(0,1,0)^T$ and $(0,0,1)^T$. The corresponding y values would immediately give the A coefficients. A similar arguments holds for any set of three x samples that are linearly independent. Note that the matrix A can be identifiable even if it isn't invertible, since we are allowed to probe it with multiple (x, y) pairs.

For questions of identifiability, it is usually assumed that we have the correct model, so that with the true A we would match the data exactly.

20: Reflecting our biophysical application, we will sometimes refer to these as *compensatory effects* so as to also include cases where we don't have perfect invariance of the model under certain parameter transformations, but nevertheless it is only weakly sensitive to such transformations.

Important

A model being *correct* (aka "true") does not make it *identifiable*.

This has important practical implications, because a procedure that attempts to fit a non-identifiable model will likely never converge. Instead we want to identify and remove parameter symmetries—in this example, we would replace the parameters in Equation (1.25) with their sum $A' := A + B$. In addition to making the problem tractable, identifying such symmetries can provide valuable insights into the interactions between parameters.

It is also worth noting that models can be *technically* identifiable, but *effectively* non-identifiable. For a simple example, consider data y generated from x with the law

$$y = ax + bx^2 + \xi\,, \tag{1.26}$$

with $a = 1$, $b = 10^{-6}$ and ξ a Gaussian random variable with unit variance. If the x in the data are of order 1, the true model will be mostly indistinguishable from one with $b = 0$.[21]

Effective non-identifiability is common in complex models like the aforementioned pyloric circuit, where different parameters can have compensatory effects [22, 24]. We illustrate this in Figure 1.10, which shows the posterior[22] for some parameters of the pyloric model under condition described in [24]. Bright colours indicate regions of plausible parameters: the data don't support assigning more precise values to these parameters.

As we will see in (Section 4.2.2 (p. 67)), restricting a learning procedure to identifiable parameters can be essential for that procedure to converge.

21: In theory, with enough data, one could tease out the value of b by averaging. However this only works if the model matches the data-generating process *exactly*, up to the unknown parameters.

22: The notion of a posterior density is defined in Section 2.2 (p. 35).

1.3.1. Time-series provide a lot of data

When fitting static models, the complexity of the model must be balanced with the availability of data: more complex models typically have more free parameters and require more data to identify them all without overfitting.

One of the ways in which fitting dynamical models differs from static ones is that very often we are flooded with data. Consider for example that it is not uncommon for an electrode to record at 1 kHz, and for an experiment to last a dozen minutes or more. Even just one minute of recording would provide 60,000 data points per electrode, per experiment; although the data points will exhibit correlations, this is still a lot of data to help filter out noise and resolve parameters.

In this thesis we are concerned with learning dynamical models, so that it makes sense to assume availability of unlimited data.

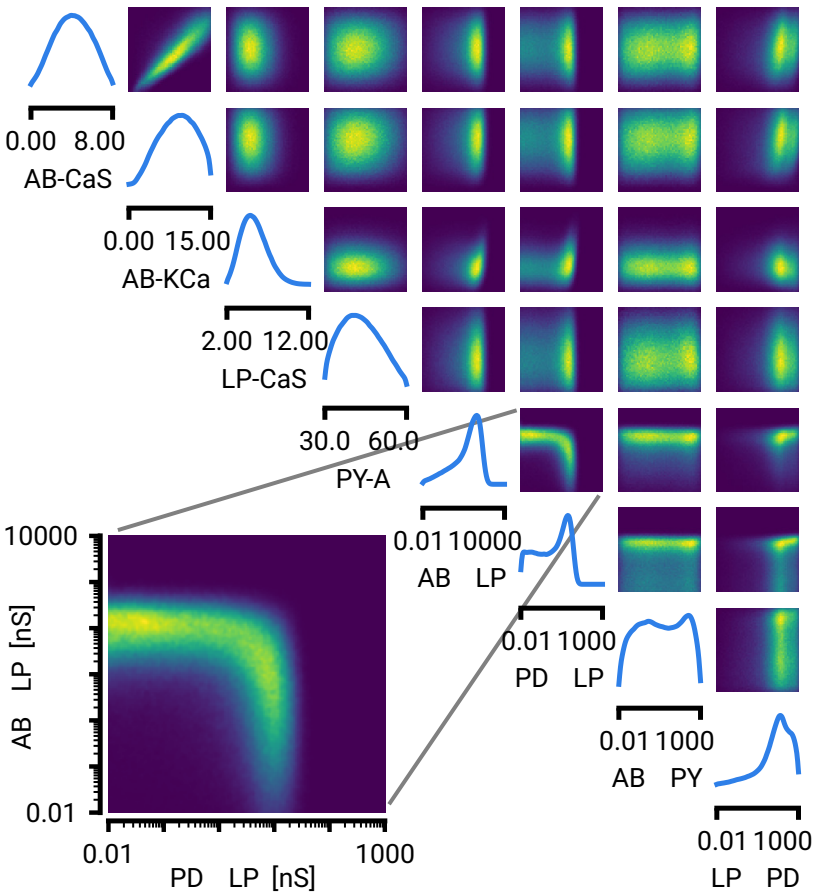


Figure 1.10.: Posterior densities for a subset of parameters of the pyloric model described in Section 1.2 (p. 15). Bright colours indicate more likely parameter combinations; The highlighted panel shows the distributions for two parameters determining the strength of connections, one from AB to LP and the other from PD to LP neurons. Adapted from Fig. 5 of [24].

1.4. Modelling populations of neurons

1.4.1. Simplifying the neuron model

When modelling populations of neurons, it is common to use much simpler models for the individual neuron. A big reason for this is lack of identifiability: while simulating thousands of conductance neurons may push computational limits,[23] determining tens of thousands of conductance parameters is simply not possible with today's data.

There is also quite a bit of room for simplification, because at the population scale, the *shape* of action potentials is not so crucial — much less than the *time* at which it occurs. Indeed, since the shape of an action is largely stereotyped, it does not need to be modelled directly: once the runaway threshold is crossed,[24] two things occur:

- The membrane potential at the soma follows a mostly pre-determined temporal trajectory—an action potential, like the one shown in Figure 1.5—and after a few milliseconds stabilizes to a subthreshold value.
- The generated action potential travels down the axon at a fixed speed, eventually triggering a response in downstream neurons (see Figure 1.1).

Most relevant therefore for population models are the subthreshold dynamics that precede the crossing of the runaway threshold.

23: Especially with the use of graphic cards, modern computers can easily integrate ODEs with thousands of dimensions.

24: Recall the phases on an action potential (p. 12).

Recall

We refer to action potentials as *spikes* when we don't care about their temporal shape, only their timing.

Of course, action potentials are not *completely* stereotyped, and how different neuron models account for this is a large part of what differentiates them. Chief among these are the *refractory period* and *adaptation* effects we mentioned in Section 1.1.5 (p. 13).

Simplified models usually aggregate multiple parameters into one *effective* parameter—often these are interpreted as average parameters [8, §12.2.2], although they may also take into account nonlinear effects; we will see an example in Chapter 4 (p. 63). Not only does this reduce the number of parameters, and thus increase the available statistical power to estimate those that remain, but typically we aggregate parameters with similar effects—i.e. exactly those with approximate symmetries under which the model is invariant. Reduced models are therefore much more identifiable.[25]

For a modeller, simpler models are generally better: they have fewer issues with identifiability, are easier to interpret, are faster to simulate and tend to generalize better to new data. For these reasons, we usually aim for the simplest model we can get away with, which nonetheless preserves enough of the essential dynamics to maintain predictive power.[26]

In the next subsections we describe a few reduced models that will appear in later chapters. For a more proper, albeit necessarily still incomplete, survey of population models, see for example [8].

25: Although identifiability is often treated as a binary (a model is identifiable or not), we say "more" identifiable here because in practice this is more nuanced. As mentioned above, a technically identifiable model can be non-identifiable in practice. Conversely, some forms of non-identifiability can be easier to resolve. A simplified model may be non-identifiable, but often can be *made* identifiable through suitable reparametrization, in contrast to a large model with tens of thousands of parameters.

26: What is meant by "predictive power" will depend strongly on the application. In some cases quantitative features may be enough, while in others we may seek to match some measurements exactly. The amount of simplification we can get away with will vary accordingly.

1.4.2. Leaky Integrate-and-fire (LIF)

The *leaky integrate-and-fire* (LIF)—formalised by Stein [25] and later by Knight [26][27]—takes the idea of simplifying membrane potential dynamics to the extreme. The dynamics of an individual neuron i are reduced to a single variable, the membrane potential u_i, which accumulates inputs (weighted by some abstract resistance R to keep units consistent):

$$\tau_i \frac{du_i}{dt} = -u_i + u_{\text{rest}} + RI^i_{\text{input}} + RI_{\text{ext}} \quad \text{if } u_i < \vartheta\,. \tag{1.27}$$

The leak term $-u_i$ represents the membrane returning to its resting potential u_{rest} in the absence of inputs I_{input} from other neurons. (Or, in the circuit model, to the capacitor discharging at rate $\tau = {}^{C}/_{g}$.)

As we motivated in Section 1.4.1, communication between neurons depends only on the times at which spikes are emitted; this is reflected in the input I^i_{input}, which consists of a train of spikes coming from other neurons:

$$I^i_{\text{input}} = \sum_{j \in \mathcal{I}_{-i}} \sum_{t^j_k < t} w_{ij} \delta(t - t^j_k)\,, \tag{1.28}$$

where $\mathcal{I}_{-i}$ is the set of neuron indices excluding i and the t^j_k are the times at which other neurons have emitted a spike. In this way, an LIF neuron "integrates" its inputs, and if sufficiently many spikes arrive in a short enough period of time, the membrane potential crosses a threshold ϑ. The matrix w encodes the connectivity of the network: which neurons are connected, in which direction, and how strongly.

Notation

It is common in population models to denote the membrane potential as u or h instead of V, since it is more of an abstract accumulation variable than a true membrane potential. Currents like I^i_{input} are similarly more abstract, and don't correspond as measurable ionic currents.

Remark

In Equation (1.18) (p. 12) we model changes in *current*, while in Equation (1.27) we model changes in an abstract *voltage*. The former is more easily compared to actual measurements, while the latter is easier to make dimensionless (and thus study theoretically).

Note

To simplify the exposition, we will omit the term RI_{ext} denoting external input in the equations that follow. Mathematically it simply shifts the resting potential u_{rest}.

27: The integrate-and-fire model is often credited to Louis Lapicque (1907), even though his study is not actually about integrate-and-fire models; see Brunel and van Rossum [27] for a discussion of that paper and an overview of the history of the LIF model.

It is convenient to define the *spike train* emitted by neuron i as

$$s_i(t) = \sum_{t_k^j < t} \delta(t - t_k^j) \tag{1.29}$$

so that the input reduces to

$$I^i_{\text{input}} = \sum_{j \in \mathcal{J}-i} w_{ij} s_j(t)\,. \tag{1.30}$$

In this model, the initiation of the runaway process that initiates a spike is reduced to a voltage threshold ϑ; when it is exceeded,

$$u_i \geq \vartheta\,, \tag{1.31}$$

two things happen: 1) a spike is "fired", and 2) the membrane potential u_i is immediately set to a fixed reset potential u_{r}. The entire model can therefore be written

$$\tau_i u_i(t + dt) = \begin{cases} \tau_i u_i - (u_i - u_{\text{rest},i})dt + dt\, R \sum_{j \in \mathcal{J}_{-i}} w_{ij} s_j(t) & \text{if } u_i < \vartheta \\ u_{\text{r},i} \quad (\&\text{ emit spike}) & \text{if } u_i \geq \vartheta\,. \end{cases} \tag{1.32}$$

It is important to remember that in this model, u_i is more of an abstract accumulation variable than a true membrane potential—we have simply thrown too many details away to hope to match the precise values of u_i one might record in an experiment. Only the spike times, or at least their statistics, have some hope of being comparable with experiments.

1.4.2.1. Making LIF neurons identifiable

Notice however that if we only observe spikes, the model given in Equation (1.27) is *non-identifiable*. Indeed, the sequence of emitted spikes is completely unchanged if we simultaneously shift the reset potential and the threshold:

$$u_{\text{r}} \leftarrow u_{\text{r}} + \Delta \qquad \vartheta \leftarrow \vartheta + \Delta\,. \tag{1.33}$$

So it is not possible to learn both u_{r} and ϑ.

Similarly, we also preserve the output under a simultaneous rescaling of τ and ϑ, so it is not possible to learn both of those parameters either.

For these reasons, we can set $u_{\text{r}} = 0$ and $\vartheta = 1$ without loss of generality. Working with such a reduced model also ensures that the remaining parameters are identifiable.

Note that the same remark applies to the weights: if there are N neurons, then of the $N^2 + 1$ parameters $\{w_{ij}\}_{i,j=1}^N \cup \{R\}$, only N^2 are identifiable. This case requires a bit more care: we don't want to parameterize one population differently from the others, as that tends to make inference more difficult. One solution is to fix R to 1; another is to fix the norm of the w matrix, although this comes with additional challenges.[28]

28: While there are advantages to factoring out a global input strength scale R for learning, these are often negated by the additional challenge of constraining learning of w to a subspace of all matrices. For example, if we limit the w to orthogonal matrices and parametrize them by angles, then not all angles are equivalent.

1.4.3. Grouping neurons into populations – Excitatory-inhibitory (E-I) networks

While it is usually not feasible to assign separate parameters to every neuron, we *can* classify them into broad categories; the neuron types defined in Section 1.2 (p. 15) are an example of this. Often these types can even be visually distinguished; for example, the "pyramidal cell" in Figure 1.11 has a long axon that projects down into the cortex, while the axons of the neighbouring "basket cell" project more horizontally (i.e. they remain within a cortical layer). Although it is not always easy or possible to assign a cell type based solely on a cell's morphology, generally we can do so if we know how it responds to inputs, so it makes sense to include in our models different functional populations with different parameters.

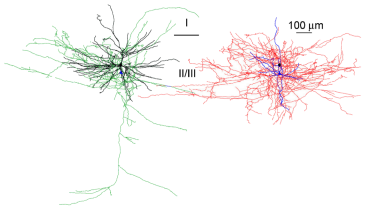


Figure 1.11.: Excitatory and inhibitory neurons can be distinguished under the microscope. Optical microscopy reconstructions of human excitatory pyramidal cell (left) and inhibitory basket cell (right). Soma and dendrites are coloured coloured black (left) or blue (right). Axons are coloured green (left) or red (right). Scale bars apply to both cells; "I" and "II/II" indicate the cortical layer. Adapted from [28].

It turns out that, to a large extent, neurons satisfy something called "Dale's principle" [29]: they tend to emit only one type of neurotransmitter, and therefore form only one type of connection.[29] This principle motivates what is likely the simplest and most common classification of neurons, as either *excitatory* (E) or *inhibitory* (I) cells: excitatory neurons increase the propensity of downstream neurons to fire, while inhibitory neurons do the opposite. In an LIF model, this is represented by the sign of the connectivity weights w_{ij} in Equation (1.32): a positive (resp. negative) value of w_{ij} means that the connection from neuron j to i is excitatory (resp. inhibitory), since the spikes emitted from j cause an increase (resp. decrease) in the membrane potential of i.

29: As far as biological principles go, Dale's principle is uncharacteristically robust, to the point that it is often presented as a universal, inviolable rule by theorists [8, §16.3.2]—imposing Dale's principle is even a frequent manner in which artificial neural networks are made "biologically plausible" [30–33]. Of course, real biology is much more messy: although the principle remains broadly valid and a very useful simplification, it is not that uncommon to find neurons emitting multiple neurotransmitters, sometimes even both excitatory and inhibitory [34–36].

A common way to write Equation (1.32) for an E-I network is therefore to split the sum over the two populations, so that the dynamics of a neuron i in population $\alpha \in \{E, I\}$ are given by

$$\tau_\alpha u_i(t+dt) = \begin{cases} \tau_\alpha u_i - (u_i - u_{\text{rest},i})dt + dt \sum_{\beta\in\{E,I\}} \sum_{j\in\mathcal{J}_\beta} w_{ij}^{\alpha\beta} s_j(t) & \text{if } u_i^\alpha < \vartheta^\alpha \\ u_{\text{r},i} \quad (\&\text{ emit spike}) & \text{if } u_i^\alpha \geq \vartheta\,. \end{cases} \tag{1.34}$$

Here $\mathcal{J}_\beta$ is the set of indices for neurons in population β, and it is understood that

$$w_{ij}^{EE} \geq 0 \qquad w_{ij}^{EI} \leq 0 \tag{1.35}$$

$$w_{ij}^{IE} \geq 0 \qquad w_{ij}^{II} \leq 0\,. \tag{1.36}$$

This is often schematically represented as in Figure 1.12.

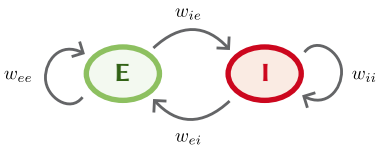


Figure 1.12.: Representation of an E-I network with four types of connectivities: w_{ee}, w_{ei}, w_{ie} and w_{ii}

One issue with population models is that biology is messy: unlike atoms or molecules, no two neurons are identical, and treating them as such—i.e. giving each identical parameters—can produce unrealistic synergies in the population activity. We can avoid this by drawing the parameters from random distributions; for example, we might have four separate distributions for the connectivities $w_{ij}^{\alpha\beta}$, to represent the four types of connectivities $E \leftarrow E$, $E \leftarrow I$, $I \leftarrow E$ and $I \leftarrow I$.

Terminology

We say that a population is *homogeneous* if all neurons it contains share the same parameters, and *inhomogeneous* if each neuron has different parameters.

1.4.4. A more realistic LIF: the generalized integrate-and-fire (GIF)

The LIF model is rather crude, and in particular ignores the fact that neurons exhibit both *adaptation* and *stochasticity*.[30] By extending the model with a soft, adaptive threshold, we obtain the *generalized integrate-and-fire* (GIF) model, which captures enough of the key biological features to accurately predict spike times, while still remaining mathematically tractable [8, 37, 38].

The GIF model accounts for adaptation by allowing the firing threshold to vary over time: conceptually, each emitted spike temporally increases the threshold, which eventually decays back to its baseline $u_{\text{th},i}$ in a stereotyped fashion (see Figure 1.13). This stereotyped response can modelled as a convolution kernel θ_i,

$$\begin{aligned}\vartheta_i(t) &= u_{\text{th},i} + \int_{-\infty}^{t} \theta_i(t-\hat{t}) s_i(\hat{t}) d\hat{t} \\ &= u_{\text{th},i} + \big(\theta_i * s_i\big)(t)\,,\end{aligned} \tag{1.37}$$

a form that is also used in the related *spike response model* (SRM) [8, §6.4]. The second line assumes the usual case of a causal kernel, i.e. one that is zero for times in the future. We also assume that θ decays back to zero for large enough times, i.e. that the neuron eventually "forgets" its spike if we wait long enough.

In full generality therefore, an adapting threshold ϑ should depend on the entire history of spikes emitted by neuron i up to t:

$$\mathcal{H}_i(t) := \big\{t_{i,k} \mid t_{i,k} < t\big\}\,. \tag{1.38}$$

Neurons then fire based on the difference between their membrane potential and their threshold,

$$u_i\big(t\big) - \vartheta_i\big(t, \mathcal{H}_i(t)\big)\,, \tag{1.39}$$

Note that we consider homogeneous populations, where the neurons in population α share the same kernel θ_α, so that the threshold function ϑ can be indexed by α instead of i.[31]

In contrast to the LIF, in a GIF this difference is not used as-is to determine spiking, but converted to a "firing intensity" or *hazard rate*[32] $\lambda_i(t)$; for homogeneous populations, this takes the form

$$\lambda_i(t) = f_\alpha\Big(u_i\big(t\big) - \vartheta_\alpha\big(t, \mathcal{H}_i(t)\big)\Big)\,, \tag{1.40}$$

for some monotone-increasing f_α. We require that $f^\alpha(x) \approx 0$ when $u_i^\alpha \ll \vartheta_i^\alpha$ (i.e. the neuron should not fire when it is far below threshold). A common form, and the one we will use in Chapter 4 (p. 63), is an exponential:

$$f_\alpha(x) = c_\alpha \exp(x/\Delta_{u,\alpha})\,, \tag{1.41}$$

where the parameter Δ_u^α controls the "softness" of the threshold, while c_α can be interpreted as the instantaneous rate at threshold [20].

The hazard rate λ_i is an instantaneous rate, so that the expected number of

30: Whether due to the thermodynamics of ion channels, or simply because we cannot plausibly model every input from every other neuron, from the point of view of the modeller, neurons are intrinsically stochastic. Irrespective of the origin of stochasticity, the result on generated spikes is similar and can be treated with similar models [8, §2.3, §9].

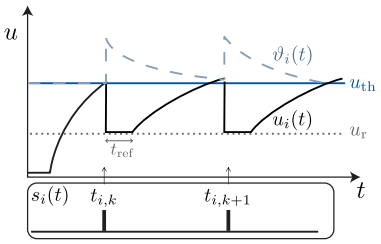


Figure 1.13.: Threshold dynamics in the GIF model. When a neuron emits a spike, its membrane potential is reset to u_r and stays there for the duration of the absolute refractory period t_{ref}. Simultaneously, the neuron's firing threshold is increased by a fixed amount, then follows a stereotyped decay back to the baseline threshold u_{th}. The lower box shows the spikes emitted by the neuron; its inputs are not shown. Note that the time scales were chosen for illustrative purposes: ϑ rarely relaxes completely, since its timescale is typically much longer than the interspike interval. Adapted from figure 6.12 of [8].

Note

One of the main drivers of adaptation are channels for other ion species, like the Ca^{2+} channels described in Section 1.2.1 (p. 17), which allow changes in ion concentrations to accumulate over many spikes. For example, Ca^{2+} ions may rush in alongside K^+ ions during a spike (e.g. through their own high-threshold channels), but have comparatively slower recovery dynamics [see 8, p. 48]. This is essentially what we model with Equation (1.37). The fact that those complex channel dynamics can be reduced to a simple response filter, while still retaining the ability to predict spike times, is indicative of the effectiveness of the response model.

31: The actual value of the threshold of course still depends on each neuron's individual history, so that we have $\vartheta_i(t) := \vartheta_\alpha(t, \mathcal{H}_i(t))$.

32: The term *hazard rate* comes from actuarians, who were among the first to study the occurrence of discrete events (like a fatal accident) occurring in continuous time.

spikes within an interval $[t, t+\Delta t]$ is

$$\mathbb{E}_{[t,t+\Delta t]}\big[\#\text{ spikes}\big] = \mathbb{E}\left[\int_t^{t+\Delta t} s_i(t)ds\right] = \int_t^{t+\Delta t} \lambda_i(s)ds\,. \tag{1.42}$$

In this sense it is conceptually analogous to the instantaneous rate of a Poisson process, with the key difference that here successive events are not independent since each spike changes the firing threshold.

1.4.4.1. Renewal vs non-renewal neurons

A *renewal process* is one where the waiting time for r events can be written

$$T_r = X_1 + \cdots + X_r\,, \tag{1.43}$$

where the X_i are independent and identically distributed [39, p. 25]. The X_i here are random variables describing the waiting time for one event. A constant hazard rate produces waiting times that are exponentially distributed, and if we make the X_i i.i.d. exponential random variables, we recover the Poisson process.

A renewal process is *Markovian* (i.e. "memoryless"): predicting the time of future events only requires knowledge of the most recent one. For this reason, it is popular in theoretical neuroscience to approximate neurons as renewal processes, since it allows one to identify them by the time $\hat{t}$ of their last spike. A sum over neurons can then be replaced by an integral over spike times, where instead of tracking the entire population of neurons, we can instead track a density ρ over spike times:

$$\sum_i \langle\cdots\rangle \leftrightarrow \int_{-\infty}^{t} \langle\cdots\rangle\, \rho(\hat{t})\, d\hat{t}\,. \tag{1.44}$$

This allows one to obtain closed-form or self-consistent equations for a number of quantities; see for example the spike response model (SRM) of Gerstner [40].

Of course, we know that neurons are *not* memoryless: their firing intensity depends on the entire history of their spikes, not just the most recent one. Predicting accurate spike times with a renewal process is not really possible, which is why Equation (1.37) integrates the adaptation kernel θ over all past times $\hat{t}$.

Nevertheless, the GIF model is in a sense *almost* renewal, since the membrane potentials are always reset to the same value, so that the only source of non-Markovianity is the adaptation threshold ϑ. Moreover, Equation (1.37) is somehow too detailed: after some time, the effect of a spike will be forgotten (i.e. its affect on relative refractoriness will have decayed), and keeping it in the integral just muddies the mathematics and makes computer simulations more onerous. A key insight of Naud and Gerstner [41] is that in many cases,[33] we can in fact only track the time of the most recent spike, as long as we also track the history of the aggregate population activity.

Intuitively, this is based on two ideas:

- it becomes less important to remember exactly when spikes occurred the farther back in time they occurred; and

33: The most obvious case where a quasi-renewal approximation would *not* be appropriate is an intrinsically bursting neuron, which say tends to fire 4 spikes in rapid succession. To model such bursts accurately, a history of at least 4 spikes would be needed.

- since neurons from the same population have similar firing histories, we can use the population activity as a proxy to estimate the current threshold of each neuron.

The formalization of this intuition leads to the *quasi-renewal theory* of Naud and Gerstner [41].

1.4.5. Aggregating populations with the mesoscopic GIF

One of the key features of the GIF model is that it is accurate enough to predict spike times of individual neurons. In many cases however, this kind of precision is unnecessary: on the one hand, it is very difficult to interpret the activity of tens of thousands of neurons without some kind of averaging, and on the other, many experimental methods don't provide that resolution anyway.[34] Many questions also concern only the population activity. For example, pathologies like Parkinson's or epilepsy are also diagnosed based on the population activity; if the aim is to understand how the pathological population activity emerges, it is better to model populations directly—rather than first simulating a super-resolved model and then averaging across populations—for the same reasons as before: simpler models are easier to interpret, more identifiable, and more efficient to simulate. We will qualify as *microscopic* a model that tracks the membrane potential of each individual neuron.

Population models are by design lower dimensional than microscopic ones. But they often come with the tradeoff of less accurate representations: the interactions between all those neurons leads to highly non-trivial statistical correlations, both between populations and between points in time, which are difficult to describe.

This complexity is well illustrated by the work of Schwalger, Deger, and Gerstner [20], who derive equations for the activity of finite *mesoscopic*[35] populations of GIF neurons. Specifically they do the following. Given the following definition of population activity,

$$A_\alpha(t) = \sum_{i \in \mathcal{I}_\alpha} s_i(t) \tag{1.45}$$

they derive a closed update formula F,

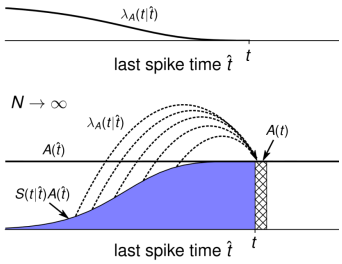
$$A_\alpha(t + \Delta t) = F(t, \{\mathcal{H}_\beta(t)\}_\beta)\,, \tag{1.46}$$

where $\mathcal{H}_\alpha(t) := \{A_\alpha(\hat{t})\}_{\hat{t}<t}$ is the history of past activities for population α. This allows us to update the population activities A_α using only knowledge of the A_α themselves—F is allowed to depend on the history of population activities, but not on the spike trains of individual neurons. Note that since each neuron is stochastic, for a finite population of N neurons, $A(t)$ will also be stochastic, albeit with decreasing variance as we increase N.

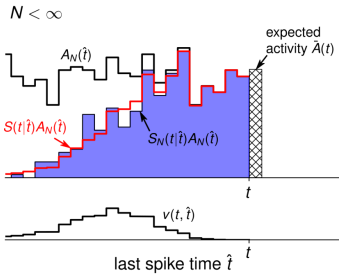
The key assumptions in the derivation are:

Homogeneous populations All neurons within one population have exactly the same GIF parameters.

Implicitly, this assumes that we can determine to which population each neuron belongs. This is easy to do in simulation, but since populations can be co-located in the brain, it is non-trivial (although still plausible) to do this in experimental data.[36]



(a) When the number of neurons in a population is infinite ($N \to \infty$), the hazard rate $\lambda(t \mid \hat{t})$ is equal to the fraction of neurons of "age" $t - \hat{t}$ that will fire at t. These fractions add up to form the total population activity $A(t)$ at t. The hazard typically increases with age, as shown at the top.



(b) When each population contains only a finite number of neurons ($N < \infty$), the hazard rate is only an approximation of the fraction that fire at each time t. This leads to additional finite-size fluctuations in the activity $A(t)$. The mesoGIF model introduces a random variable $S(t \mid \hat{t})$ representing the fraction of neurons that have last fired at $\hat{t}$; the variance of that random variable is $v(t, \hat{t})$.

Figure 1.14.: Generation of spikes in the mesoGIF model. Panels reproduced from [20] (Fig. 2).

34: Since a brain has $\sim 10^{10}$ *neurons*, any experimental method with single-cell resolution must either focus on a single very small area, or record only a handful of cells from multiple areas. A useful recording of spike times also needs sub-millisecond precision. Notwithstanding the technical challenge of probing so many neurons, simply storing the huge amount of data becomes a real problem.

35: The term *mesoscopic* distinguishes these models from both *microscopic* ones, where each individual neuron is resolved, and *macroscopic* ones, where the assumption of homogeneous populations is hard to justify.

This limits the size of a population. Schwalger, Deger, and Gerstner [20] suggest populations of a few thousand, based on the cortical column model of Potjans and Diesmann [42].

Absolute refractory period This is not so much an assumption—we know that neurons have a period of ~ 1 ms after a spike during which they effectively ignore inputs inputs—but it justifies formulating the model in terms of **discrete time bins** Δt, where Δt is at most the absolute refractory period t_{ref}.[37]

Relative refractory period After the absolute refractory period has expired, a neuron does not immediately become fully active; instead its firing intensity gradually increases back to a "free" firing intensity. (This is illustrated at the top of Figure 1.14a, and corresponds to the shrinking difference between u and ϑ in Figure 1.13.)

In particular this means that neuron spikes cannot be treated as a renewal process, since they retain memory of their previous spike.

Quasi-renewal approximation This approximation allows us to identify neurons by the time of their last spike, thus recovering some of the mathematical tractability of renewal neurons. Since we have discretized time, the density over spike times $\rho(\hat{t})$ forms a histogram.

The quasi-renewal approximation means that we can replace dependence on a neuron's history in Equation (1.40) (p. 24) with dependence only on the time $\hat{t}$ of its last spike, and on the history of population activity:

$$\lambda(t \mid \hat{t}) = f_\alpha\Big(u(t, \hat{t}) - \vartheta_\alpha(t, \hat{t}, \mathcal{H}_\alpha(t))\Big)\,. \tag{1.47}$$

We can now apply Equation (1.44) (p. 25) to represent the state of a population by a large vector S, where $S(t \mid \hat{t})$ is a *survival function*: the fraction of neurons at time t whose last spike occurred at $\hat{t}$. Since each neuron belongs to exactly one bin, integrating over past spike times is the same as summing over neurons:

$$\bar{A}_a(t) \;:=\; \mathbb{E}\big[A_\alpha(t)\big] \;=\; \int_{-\infty}^{t} \lambda_\alpha(t \mid \hat{t}) S_\alpha(t \mid \hat{t}) A_\alpha(\hat{t})\, d\hat{t} \tag{1.48}$$

If population sizes were infinite, A_a would always be equal to its expectation, and $\lambda(t \mid \hat{t})$ would give exactly the fraction of neurons from bin $\hat{t}$ that fire at time t. (Figure 1.14a). The key contribution of [20] is to treat the case of *finite* populations, where $A_\alpha(t)$ and $S_\alpha(t \mid \hat{t})$ become random variables (Figure 1.14b).

This S is itself a random variable, but we can track its variance, allowing us to compute the expectation $\bar{A}_\alpha(t)$ by integrating over all possible S histories. The result is a set of closed equations for three large coupled state vectors: the membrane potential $u(t \mid \hat{t})$, the survival fraction $S(t \mid \hat{t})$, and the variance on that fraction $v(t \mid \hat{t})$.

We will refer to these equations as the *mesoscopic GIF* (mesoGIF) model, and will discuss them in more detail in Chapter 4 (p. 63), where we study how this model might be learned by gradient descent.

Note

In practice we truncate the integral in Equation (1.48) to exclude neurons whose last spike time is so long ago that it has effectively been forgotten.

36: At a minimum, any section tissue would need to be modelled with two populations, since excitatory and inhibitory cells are almost always intermingled. Note also that we are speaking here of identifying neurons from a recording of possibly thousands of units, so the method would need to work at scale.

37: Some later work by [43] extends the theory to continuous time (see also [44], [45]), making the formalism less dependent on this discretization.

These "free" neurons are treated as their own separate time bin; if we denote their number by N_{free}, and the forgetting time by T, Equation (1.48) can be rewritten as

$$\begin{aligned} \bar{A}_a(t) = &\quad \lambda_{\alpha,\text{free}}(t) N_{\alpha,\text{free}}(t) \;+\; \int_{t-T}^{t} \lambda_\alpha(t|\hat{t})\; S_\alpha(t|\hat{t})\; A_\alpha(\hat{t})\, d\hat{t} \\ N_{\alpha,\text{free}}(t+\Delta t) = &\Big(1 - \lambda_{\alpha,\text{free}}(t)\Delta t\Big) N_{\alpha,\text{free}}(t) \\ &+ \Big(1 - \lambda_\alpha\big(t|t-T\big)\Delta t\Big)\; S_\alpha\big(t|t-T\big)\; A_\alpha\big(t-T\big)\; \Delta t\,. \end{aligned} \tag{1.49}$$

In this way we only need to track finite finite vectors for λ, S and A, and the equations become closed in a computational as well as theoretical sense.
(The second equation is most easily obtained by discretizing the integral, replacing the neuron densities by finite counts:

$$\bar{A}_a(t) \approx \lambda_{\alpha,\text{free}}(t) N_{\alpha,\text{free}}(t) \;+ \sum_k \lambda_\alpha^{(k)} N_\alpha^{(k)} \tag{1.50}$$

$$N_\alpha^{(k)} = \int_{t-k\Delta t}^{t-(k-1)\Delta t} S_\alpha(t|\hat{t})\; A_\alpha(\hat{t})\, d\hat{t} \tag{1.51}$$

$$\approx S_\alpha(t|t-k\Delta t)\; A_\alpha(t-k\Delta t)\Delta t\,. \tag{1.52}$$

The appearance of Δt twice in the update equation for $N_{\alpha,\text{free}}$ reflects the double use of time, as both an indexing variable, and the dynamic variable. Note that λ has units of seconds^{-1}, while A has units of # neurons (or spikes) / second, so the units are indeed consistent.)

Interpretability of the mesoGIF parameters

A key feature of the mesoGIF model is that each of its parameters describes characteristics of single-neuron dynamics: the instantaneous rate at threshold, the reset potential, the synaptic time constants, etc. When we fit it to data in Chapter 4 (p. 63), we are in effect asking the question "assuming homogeneous populations, what single-neuron parameters would produce this aggregated population activity?".
Through the homogeneous assumption, the model thus provides a tight link between the microscopic and mesoscopic dynamics; between a plausible *implementation* at the single neuron level, and its *manifestation* at the level of populations.

2. Learning models by optimization

One distinguishing feature of the neuron models we presented in Modelling neurons is their complexity: as we discussed in Section 1.3 (p. 18), the large number of parameters—and compensatory effects between them—conspire to make those parameters difficult, if not impossible, to identify. Their dynamic diversity—what we may also describe as their *expressivity*—also makes exploring the parameter space difficult, since the sensitivity to parameters is usually non-trivial and often highly non-uniform—especially in the vicinity of bifurcations.

That being said, at least the stochastic models* we will consider can still be viewed as statistical models, producing observations according to some probability distribution. The difference with the models one usually encounters in the statistics literature is that here that distribution is defined only *implicitly*, through a set of stochastic dynamical equations.†

Correspondence of dynamic and statistical models

Stochastic dynamical models typically have the form[a]

$$q(y_t \mid y_{t-\Delta t}; \theta)\,, \tag{2.1}$$

meaning that the probability of an observed realization can be written

$$q(\{y_0, y_{\Delta t}, \cdots, y_{L\Delta t}\} \mid \theta) = q(y_{\Delta t} \mid y_0; \theta) \cdots q(y_{L\Delta t} \mid y_{(L-1)\Delta t}; \theta)\, q(y_0 \mid \theta) \tag{2.2}$$

The same kind of unrolling of the probability can be done more generally for non-Markovian and/or continuous-time systems, as long as they are *causal* (they do not depend on the future); the idea is the same, but the notation heavier and more technical.

The ability to interpret dynamic models as statistical models (i.e. as probability distributions) underlies much of our methods.

[a] For simplicity, we use the form for a Markovian set of equations with discrete time steps of length Δt: y_t is the state at time t and θ is a vector of model parameters.

Note

In this thesis we consider stochastic equations that converge to some stationary distribution when averaged over sufficiently long time, and therefore have an unambiguous corresponding statistical model.

Another distinguishing feature of those models is their *interpretability*: each parameter, like the conductance (p. 9) of an ion channel or an neuron's refractory period (p. 13), is tied to a specific, simpler mechanism, and we understand the neuron's full dynamics as the combination of these mechanisms. Albeit imperfect, this deconstruction into simpler mechanisms is what allows us to hypothesize how the dynamics would change under different conditions, and then design experiments to test those hypotheses.

p vs q probabilities

We follow the convention that p is a true probability, whereas q is a probability given by some model.

* The GIF model (p. 24) is one example of an intrinsically stochastic model, but even deterministic models are often augmented with additive noise to account for measurement uncertainty.

† Strictly speaking we will mostly consider *update equations* that are discretizations of a continuous time process. In particular the mesoGIF model (Section 1.4.5 (p. 26)) is more naturally expressed as an update equation.

Science depends on interpretability

It is thanks to models being *interpretable* that fitting them to data can inform the next generation of experiments, and thus advance our broader scientific understanding.

In the following sections, we start by introducing the framework of *empirical risk minimization* (ERM), which is formalized within *learning theory*; this is the theory that underlies modern machine learning. We also discuss why this is an appropriate framework for learning interpretable models. We then broaden the scope and see how Bayesian models can quantify the uncertainty on inferred parameters. Finally, we close this chapter by looking at how the computational technology developed for training artificial neural networks has enabled novel methods for solving the learning problem with neuron models.

Overlapping inference terminology

While "inferring parameters" and "training a model" may be mathematically equivalent, they communicate different assumptions and intentions, inherited from separate research traditions.
We list here some of the most common synonyms to "fitting", which may appear at different points in this thesis.
Fitting Simple statistical models, like a Poisson distribution or a line in linear regression, are usually said to be fitted to data. This usually implies a procedure that is *deterministic, cheap and reliable, so that the ultimate fitted model does not depend on the fitting procedure.*

In some cases closed form analytical solutions are available, such as ordinary least squares for regression and maximum likelihood estimates for some statistical distributions.
Learning is conventionally used for fitting procedures that are iterative, especially those based on gradient descent.

This term is most closely tied to statistical learning theory and machine learning.
Training is more or less exclusively used for gradient descent procedures, especially those employing large datasets and long training times that can run for hours or days.

This is the term of choice for black-box models like neural networks, which predict outcomes but are not otherwise interpretable.

Training procedures are often sophisticated and partially stochastic; *in contrast to "fitting", a trained model usually strongly depends on the training procedure.*
Inference is used when fitting a model with interpretable parameters, such as the neuron models of Chapter 1 (p. 6).

It indicates an interest in the parameters themselves, rather than just the obtention of a predictive model, so as to interpret them.

This is contingent on the model (and therefore the parameters) being close enough to the true data-generating process to be informative.

2.1. Fitting by minimizing the risk

2.1.1. A model defines a hypothesis space

In machine learning, a *hypothesis space* is the space of all hypotheses we are willing to use to explain our data [46, pp. 66, 72]. For example, we might consider a hypothesis space spanned by the parameters τ and w of an E-I network of LIF neurons, as defined in Section 1.4.2 (p. 21) and Section 1.4.3 (p. 23):

$$\mathcal{H} = \left\{ \mathcal{M}(\tau, w) \,:\, \tau \in \mathbb{R}_+^M,\, \omega \in \mathbb{M}_{2M\times 2M}(\mathbb{R}_+) \right\}, \tag{2.3}$$

where M is the number of neuron populations and $\mathcal{M}(\tau, w)$ denotes the Wilson-Cowan model parameterized by the timescale τ and connectivity matrix ω. Note that we place hard constraints these parameters by restricting them to positive numbers, since a) timescales are always positive and b) the sign of the connectivities are already determined by the type of a population (i.e. whether it is excitatory or inhibitory).[1]

A different hypothesis space may be the result of assuming a different model, different restrictions on its parameters, or both.

Both in this thesis and in the broader literature, the word *model* may be used to mean two things:

- either a set of equations with *undetermined parameters*, i.e. a *hypothesis class*;[2]
- or set of equations *along with* values for its parameters, a.k.a. a *fitted model* or a hypothesis.

When the meaning is not clear from context, we will use *hypothesis class* or *fitted model* to avoid ambiguity.

Terminology: expressivity

"Expressivity" refers to a model's flexibility: more expressive models are able to fit a greater variety of datasets, but are also more prone to *overfitting*. It is a way of speaking in qualitative terms about the size of a model's hypothesis class.

As noted above, a fitted model also implicitly defines a probability distribution q. If the model is good, then it should look like the observed data $\mathcal{D}$ were drawn from this distribution:

$$\mathcal{D} \sim q(\cdot \mid \mathcal{M}; \theta) \qquad \text{(approximate)}\,. \tag{2.4}$$

2.1.2. Empirical risk and consistent learning machines

Generally speaking, fitting a model involves minimizing a loss functional over its hypothesis space.[3] In statistical learning theory, the loss is defined per sample, to represent the cost of a prediction error:

$$\begin{aligned} Q \,:\, \mathcal{D} \times \mathcal{H} &\to \mathbb{R} \\ (y_i, \theta) &\mapsto Q(y_i, \theta) \end{aligned} \tag{2.5}$$

1: We use a slight abuse of notation in Equation (2.3): between the space of connectivity *magnitudes* ω defined therein and the space of signed connectivities ω of an E-I network, some columns are made negative. But since the sign is predetermined by the population type, it does not affect the structure of the parameter space.

Note

Ensuring that hard constraints are respected when fitting the model is key to preserving its interpretability.

2: In the context of modelling, the terms *structural model* or *structural equations* may also be used to refer to the hypothesis class.

3: Although there are exceptions (Rosenblatt's perceptron [47] is one example of a learning machine that is not defined by way of an objective to minimize), to my knowledge all modern learning methods are expressed in the form of optimizing some objective. This is perhaps not surprising, if we think of how effective the energy minimizing paradigm has been in physics.

where $\mathcal{H}$ is the hypothesis space (identified with the space of parameters) and the index i is a reminder that these are individual samples. "Samples" may comprise both input and output values; for example, the common squared loss for a model f that predicts y_i from x_i might look like

$$Q((x_i, y_i), \theta) = (y_i - f(x_i; \theta))^2 . \tag{2.6}$$

The task of a learning machine [48, §1] is to select from $\mathcal{H}$ the parameters that would minimize *expected loss*:

$$\theta^* := \operatorname*{argmin}_{\theta \in \mathcal{H}} \mathbb{E}_{y \sim p}\big[Q(y, \theta)\big] . \tag{2.7}$$

Note that the expectation in Equation (2.7) is over the true probability p. Since by definition we don't have access to p, we use instead an empirical estimate of the risk:

$$\hat{\theta} := \operatorname*{argmin}_{\theta \in \mathcal{H}} \frac{1}{L} \sum_{i=1}^{L} Q(y_i, \theta) , \tag{2.8}$$

where the hat indicates that $\hat{\theta}$ is an *estimator*: a random variable that depends on a particular draw of samples y_i from the true distribution p.

Terminology: Risk

The loss functionals in Equations (2.7) and (2.8) are termed

Risk	Empirical risk
$R(\theta) := \mathbb{E}_p\big[Q(y_i, \theta)\big]$	$\hat{R}(\theta) := \frac{1}{L} \sum_{i=1}^{L} Q(y_i, \theta)$

To distinguish from the empirical risk, the former is sometimes referred to as the *expected risk*, *true risk* or *actual risk*.

Empirical risk minimization (ERM)

Empirical risk minimization (ERM) refers to minimizing the empirical risk (Equation (2.8)) with the goal of ultimately minimizing the true risk (Equation (2.7)).

In general we expect Equation (2.8) to underestimate the true risk, since it can overfit by selecting a $\hat{\theta}$ that is only good for the finite observed data. Overfitting becomes more difficult as the number of samples L increases, so that we expect $\hat{\theta}$ to converge to θ^* as $L \to \infty$ (Figure 2.1).

The Key Theorem of Learning Theory provides necessary and sufficient conditions for $\hat{\theta}$ to converge to θ^* [48, §2]. In short:

- If the event space for observations y_i is *finite*, there is no problem: convergence follows immediately from the law of large numbers. Intuitively, once we have drawn from every possible value of y_i, the fit can no longer "cheat" by concentrating probability on the observed samples.
- However, if the event space for y_i is *infinite*, then we need to restrict the size of the hypothesis space. Here, a learning machine that is allowed to learn any probability measure can always "cheat" by concentrating

Terminology: consistency

An estimator is said to be *consistent* if it converges to the desired value in the limit of large datasets. Note that this is not the same as an estimator being unbiased: in Figure 2.1, $\hat{R}_\theta$ is biased but nevertheless converges to $\inf_\theta R_\theta$.

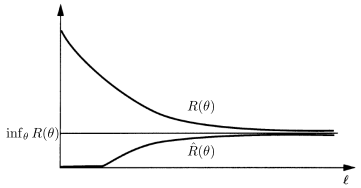


Figure 2.1.: Sketch showing an idealized convergence of both the empirical and expected risk to the infinimum of the risk (i.e. the true best fit) as the dataset size increases. (The expected risk R is putatively computed using the parameters inferred by minimizing the empirical risk $\hat{R}$.) Reproduced from [48, Fig. 2.1].

probability on observed samples, since there will always be many more unobserved ones.

Since neuron models generally predict real values,[4] we are firmly in the latter situation. Just as importantly, the fitting problems we want to solve often admit multiple solutions (one says they are *ill-posed*), and we generally want some control over which solution the learning machine selects. In both cases, the solution is to control the capacity of the hypothesis space, thus excluding (or at least rendering unfavourable) the kinds of solutions we want to avoid.

4: Even in cases where values are taken from a discrete set (due e.g. to finite neuron populations or use of a digital sensor), that set is so large that for the purposes of inference, it might as well be continuous.

Different interpretation of ERM vs. MLE, MAP

In frequentist statistics, one usually fits models by maximizing the likelihood to obtain the *maximum likelihood estimate* (MLE). If we define the loss as the negative log likelihood, then minimizing the empirical risk will yield the same parameters as maximizing the likelihood. However in ERM the loss can be defined differently, making it a more general framework. For example, defining the loss instead as the negative log *posterior* leads to ERM learning the Bayesian *maximum a posteriori* (MAP) estimate.
More importantly, in ERM the empirical risk is always interpreted as an *approximation* of the true loss, which is the expectation $\mathbb{E}_p\big[Q(y_i, \theta)\big]$. In contrast, both frequentist and Bayesian estimators are *conditioned on the observations* $\{y_i\}_{i=1}^L$ and thus treat the loss as *exact*. This leads to important differences in how the approaches define *overfitting*; in particular, in contrast to the information criteria discussed in Chapter 3 (p. 44), in ERM the notion of consistency does not rely on the true model being in the hypothesis class.

2.1.3. Choosing an appropriate loss

The formulation of ERM 2.1.2 presumes that we can quantify the cost of a prediction error with a loss functional Q. In an operational context, where a machine is trained to perform a task, this often follows naturally from that task—for example, the cost of unnecessary labour due to an incorrect diagnosis by the machine. We make two observations here: First, since each task execution corresponds to one sample, it is natural to define the minimization objective as a loss per sample. Second, the ERM framework makes no assumption on the "truth" of the model: it does not care how a model is formulated; all that matters is that it solves the task.

This is quite different from the standard maximum likelihood approach in statistics, which assumes the data-generating process to be known, modulo some unknown parameters. We can estimate those parameters by maximizing the log likelihood of the data:

$$\hat{\theta}_{\mathrm{MLE}} = \operatorname*{argmax}_{\theta} \underbrace{\log q(\mathcal{D} \mid \mathcal{M}; \theta)}_{:= \ell(\theta; \mathcal{D})} . \tag{2.9}$$

Notice that in Equation (2.9), the objective to minimize is evaluated on the entire dataset. We can obtain the type of per-sample loss used in ERM by dividing $\ell(\theta; \mathcal{D})$ by the dataset size L. Notice also that since consistency of an MLE is defined as convergence to the true parameters, it requires the true distribution to be in the hypothesis class—in sharp contrast with the definition of consistency for the ERM.

(The mathematical formulation of the likelihood should be given by the model $\mathcal{M}$.) Often it is assumed that $\log q$ has a unique maximum, in which case this estimator is consistent, in the sense that it converges to the true parameters as the number of samples L goes to infinity—**as long as the true data-generating process is one of the models within the hypothesis class.**

The log likelihood is an especially reliable objective, since it is consistent[5] even for variational problems where we learn the entire distribution p:[6]

$$p = \lim_{L\to\infty} \operatorname*{argmax}_q \log q(\mathcal{D}) = \operatorname*{argmax}_q \mathbb{E}_p[\log q]\,. \tag{2.10}$$

In fact, if we further require that an objective be *local*[7], then the log likelihood is the *only* consistent variational objective [49].

Learning interpretable models, like the neuron models of Chapter 1 (p. 6), by fitting them to experimental data lies somewhere between these situations. On the one hand, the formulation of these models certainly matters, since it provides their interpretability. We also believe that they capture some mechanisms of the true biological system. And our goal is to fit the true probability distribution as closely as possible, so using the log likelihood as an objective seems a natural choice.

On the other hand, given the approximations involved and the challenges of measuring data, we would never expect these models to *perfectly* capture the data-generating process. In this sense, the ontological frame for our problem is much better aligned with learning theory, which is designed for more flexible models which

- only approximate the true data-generating process;
- have large parameter spaces;
- have complex dependencies between their parameters leading to singularities in the likelihoods and/or multiplicities of solutions (recall Section 1.3 (p. 18));
- should generalize to new observations, *despite being imperfect*.

In light of this, we will **fit neuron models by maximizing the log likelihood** but **within the broader inductive framework of learning theory and empirical risk minimization**. Concretely this means three things:

- We divide the log likelihood by L to obtain the per-sample loss required by ERM.
- We include regularization (see Section 2.1.4 below) to reduce the hypothesis space and prevent overfitting.
- We use held-out data to evaluated fitted models, so as to obtain unbiased estimates of the expected risk.
- We rank different models according to their expected risk (see Chapter 3 (p. 44) and especially Chapter 5 (p. 99)).

2.1.4. Induction, generalization and ill-posed problems

It is important to remember that the loss landscape for the kinds interpretable models we are interested in fitting is highly non-convex: Due to compensatory mechanisms in the models, or straight-out unidentifiability (recall Section 1.3 (p. 18)), there will in general be (many) local minima reaching similar loss values. Very often these models are even *singular* [55, 56]: at the MLE, the Fisher information matrix (i.e. the Hessian of the log likelihood) is non-invertible.

This is very different from the landscapes encountered with simple statistical models, or even hierarchical Bayesian models, and much more like the loss landscape of a neural network.[8] This is one reason to frame inference within the learning theory/ERM framework.

5: "Consistency" in the sense of an objective being minimized by the true probability is more usually described as it being *proper* [49, 50]. Note for example that the likelihood (in contrast to the *log* likelihood) is *not* proper.

6: This a standard result in statistics and may be quoted in slightly different forms, among them "Gibbs' inequality" [51], the "non-negativity of the Kullback-Leibler divergence" (*ibid*), and the "consistency of the maximum likelihood estimate" [52, p. 193C].

In Equation (2.10), the maximization is done over some suitable set of probability measures, and the dataset $\mathcal{D}$ is composed of L samples drawn from the distribution p.

Remark

That the log likelihood is the only local and proper objective means that we can interpret a loss by asking what probability model it is consistent with. For example, many algorithms work by minimizing least squares ($\operatorname{argmin}\sum(y_i - \tilde{y}_i)^2$). Since the log probability of a Gaussian model is also quadratic ($-\sum(y_i-\mu)^2/2\sigma^2$), any method minimizing squared differences de facto assumes a Gaussian model for the deviations between predictions ($\tilde{y}_i$) and data (y_i). The formal equivalence of squared loss and Gaussian model is common knowledge in the machine learning literature (see e.g. [53].)

7: An objective $\ell(q,\theta)$ is *local* if it depends only on the parameters θ being evaluated, and not, say, on model predictions for some some ϵ-ball around θ—e.g. $\int_{|\theta-\tilde{\theta}|<\epsilon} q(y^n|\tilde{\theta})d\tilde{\theta}$ See [54] for a relaxation of the locality requirement by allowing dependence also on the derivatives of q, and thus accounting for a form of sensitivity to the parameters.

8: Neural networks are discussed later (p. 38) in this chapter.

Another reason is that it would make little sense to learn these neuron models if we did not expect them to be partially conserved across individuals or experiments. The logical process of going from specific training data to a more general model is called *induction*, and is closely linked to the goal of minimizing expected risk [48, §2.12]. Minimizing the expected risk (what practitioners often call *generalization error*) is central to learning theory but mostly absent from classic statistics (since the latter typically assume the true model to be within the hypothesis class).

Error = risk

Machine learning practitioners use *generalization error* to refer to the average error we expect to encounter after training a model, when we use it to predict new data. This is therefore precisely the *expected risk*. Similarly, the term *training error* is frequently used as a synonym for the *empirical risk*.

The difference between training and generalization errors is ascribed to overfitting. The practice of machine learning has converged on a few key methods to minimize overfitting and thus improve generalization:

Splitting the data into train ($\mathcal{D}_{\text{train}}$) and test ($\mathcal{D}_{\text{test}}$) datasets Models are learned by minimizing the empirical risk $\hat{R}$ on $\mathcal{D}_{\text{train}}$, but the final performance is given by evaluating it on $\mathcal{D}_{\text{test}}$.

The samples of $\mathcal{D}_{\text{test}}$ are assumed independent but identically distributed to those in $\mathcal{D}_{\text{train}}$, so that this gives an unbiased estimate of the true risk.

This does not avoid overfitting *per se*, but at least the effect of overfitting is accounted for in the final performance.

If multiple models are fitted and we need to select the one with lowest expected risk, a third *validation* set can be used: fit data with $\mathcal{D}_{\text{train}}$, select the best model according to performance on $\mathcal{D}_{\text{validate}}$, and report final performance using $\mathcal{D}_{\text{test}}$. [53, §1.3, 46, §1.4.8]

Regularization As noted above (p. 31), when the event space for observations y_i is infinite (and especially if it is continuous), we need to limit the size of the hypothesis space to prevent egregious overfitting.

The standard way to do this is by adding an additional term to the loss, to penalize certain solutions. For models with complex loss landscape, this smooths the loss landscape, favouring certain local minima and suppressing others.[9]

A common generic choice is to penalize large parameters by adding their squared norm; for example, adding this to a Gaussian model would yield[10]

$$\sum_{i=1}^{L}(y_i - \mu)^2/2\sigma^2 + \lambda\|\mu\|^2\,. \tag{2.11}$$

9: Although a regularizer does technically change the hypothesis space, it can nevertheless be seen as reducing its capacity by making it more compact [48, §4.8].

10: This is often called *ridge regression* or *Tikhonov regularization*.

Generic regularizers however work best with generic models. With interpretable models, we have the additional requirement that the parameters we learn should remain plausible. Since we generally know beforehand what kinds of values would be plausible, this is most easily done by using Bayesian priors to regularize learning.

2.2. Accounting for parameter uncertainty with Bayesian statistics

Bayesian statistics attempts to ascribe a probability to model parameters given observed data, i.e.

$$p(\theta \mid \mathcal{D}; \mathcal{M})\,. \tag{2.12}$$

When inferring interpretable models, this is in a sense exactly what we want, so it is perhaps not surprising that the Bayesian approach has become popular among modellers.

Of course, we don't have a direct way to compute $p(\theta \mid \mathcal{D})$, so we use Bayes' theorem to write:

$$p_{\text{post}}(\theta \mid \mathcal{D}; \mathcal{M}) = \frac{q(\mathcal{D} \mid \mathcal{M}; \theta)\, \pi(\theta \mid \mathcal{M})}{p(\mathcal{D} \mid \mathcal{M})}\,. \tag{2.13}$$

The r.h.s. of Equation (2.13) is composed of three terms, each probability densities:

- the *likelihood* $q(\mathcal{D} \mid \theta)$;
- the *prior* $\pi(\theta)$;
- the *model evidence* $p(\mathcal{D} \mid \mathcal{M})$.

The l.h.s. $p(\theta \mid \mathcal{D}; \mathcal{M})$, which is the probability we want, is called the *posterior*.

As noted below Equation (2.9) (p. 33), the mathematical form of the likelihood is given by the model and is therefore known. For its part, the model evidence serves only to normalize the r.h.s. so that it corresponds to a density. Therefore we can write

$$p_{\text{post}}(\theta \mid \mathcal{D}; \mathcal{M}) \propto q(\mathcal{D} \mid \mathcal{M}; \theta)\, \pi(\theta \mid \mathcal{M})\,; \tag{2.14}$$

if our goal is only to infer the parameters, e.g. by maximizing the r.h.s., we can therefore disregard the model evidence.

This leaves us with the prior $\pi(\theta)$, i.e. the probability of the parameters before seeing the data. In Bayesian statistics, the prior is intended to represent subjective domain knowledge about parameters, which works especially well with interpretable models. For example, if we want to infer the membrane timescales of Equation (1.34) (p. 23), we might use a prior like

$$\log_{10} \pi(\tau_\alpha) = \frac{1}{\sqrt{2\pi}} e^{-(\tau_\alpha - 1)^2}\,, \tag{2.15}$$

to constrain them to the vicinity of 10 ms, which is known from previous experiments to be the correct order of magnitude.

Note now that if we take the log of Equation (2.14), we have

$$\log p_{\text{post}}(\theta \mid \mathcal{D}; \mathcal{M}) \propto \log q(\mathcal{D} \mid \mathcal{M}; \theta) + \log \pi(\theta \mid \mathcal{M})\,, \tag{2.16}$$

where the r.h.s. has precisely the form of "likelihood + regularizer". **Regularized inference is therefore equivalent to Bayesian inference, with the objective interpreted as the posterior and the regularizer interpreted as the prior.** The advantage of the Bayesian point of view is that by interpreting the regularizer as a probability, it makes it easier to translate imprecise, subjective knowledge into a quantitative form.

Note also that as the number of samples increases, the relative contribution of the log likelihood to the log posterior grows (since it is in effect summed over samples). This has two practical consequences:

- We don't need to worry too much about the subjectivity of the prior, since with enough data it becomes irrelevant.

Hint

For scaling parameters (like the time scale τ), it often works best to first transform them to log space. Then the prior is effectively on their order of magnitude.

11: The least informed prior is the Jacobian prior [57]. (Paradoxically, a uniform prior is not always "minimally informative": e.g. a change of variables from θ to $\log\theta$ can transform a flat prior to a highly peaked one.) The Jacobian prior is the unique distribution for which the Bayesian model is invariant under change of variables. For more on the different roles a prior can play, and corresponding considerations for its design, see also Gelman, Simpson, and Betancourt [58].

- We can make the prior flatter[11] (a so-called "non-informative prior") to reduce its impact on inference. The trade-off is that inferring models with less informative priors requires more data.

2.2.1. Going full Bayesian: Evaluating the posterior with Markov Chain Monte Carlo

We suggested above to perform inference by maximizing the posterior; this yields the maximum a posterior (MAP) estimate. A true Bayesian however would want to use the posterior distribution itself, since in addition to the MAP, it provides valuable information on the specificity of parameters: a broad posterior indicates that the data are not particularly informative about a parameter (perhaps because it is effectively unidentifiable (p. 18)), whereas a tight posterior indicates the opposite.

Just as valuable is the possibility of exploring *correlations* between parameters, such as those illustrated in Figure 1.10, which can indicate compensatory mechanisms in the model.

The only problem with accessing the posterior is that to be interpretable, it needs to be normalized, and we can only compute the unnormalized posterior. (Normalization would require computing the model evidence, which is very difficult to do accurately.)

Thankfully, we can side-step this issue by *sampling* from the posterior, rather than computing it directly. For sampling, all we need to know are *relative* probabilities between two points in parameter space, which is exactly what we get with an unnormalized posterior. After drawing enough samples, we can recover an empirical approximation of the true posterior by converting them to histograms.

By far the most common method for sampling from a posterior, especially a complex one, is *Markov chain Monte Carlo* (MCMC). This is a procedure for producing a sequence of vectors drawn from the posterior's parameter space. Nearby samples in this sequence are correlated (hence the "Markov chain" part), but asymptotically they become i.i.d. samples of p_{post}.

Although we use MCMC briefly in Chapter 4 (p. 63), and sampling ideas more broadly are central to the method of Chapter 5 (p. 99), the theory explaining how the different MCMC algorithms achieve convergence to the true posterior, and what technical requirements they must satisfy to do this, is not required for understanding this thesis. For those we refer the reader to [59] and [60].

These days, many software packages exist that implement MCMC more or less automatically, such as PyMC [61] which we use in Chapter 4 (p. 63). This greatly reduces the need for practitioners to be familiar with the technicalities of MCMC. This is especially valuable because the more modern MCMC methods, such as Hamiltonian Monte Carlo [62, 63] , are *far* more efficient[12] but also more difficult to implement correctly. A good explanation for why Hamiltonian Monte Carlo is so much more efficient than other methods is given by [60].

Historically, Hamiltonian Monte Carlo was hampered by the need to provide analytical gradients for the likelihood and prior; this has largely been solved now with the advent of automatic differentiation, making the aforementioned packages almost[13] turn key solutions.

12: A more efficient MCMC sampler will take fewer steps before generating a new, effectively independent sample. Hamilton Monte Carlo can easily be orders of magnitude more efficient than the classic Metropolis-Hastings algorithm, which becomes unworkable on models with more complex likelihood landscapes.

13: "Almost", because while the sampler often runs out of the box, one still needs to ensure that it has reached asymptotic equilibrium. Also, even with Hamiltonian Monte Carlo, different parametrizations of the same model can have vastly different sampling efficient.

Automatic differentiation is discussed in Section 2.3.3 (p. 41).

What exactly does a Bayesian model represent?

There is an interesting ontological question, as to whether a Bayesian posterior represents *uncertainty* about parameters, or whether it represents a model where parameters are actually distributed (think population of neurons with different parameters). Modellers often think of it in the former sense, and we will do so here. It is worth keeping in mind however that, as discussed by Swigon et al. [57], the formalism sometimes assumes the latter interpretation.

2.3. Neural networks: Models designed for learnability

Deep neural networks (DNNs) are a class of models specially designed to be universal learning machines. This allows them to solve learning problems that would previously have been unthinkable, and that success has spurned the development of technology to learn ever larger and more complex DNNs.

A substantial part of our work is based on adapting that technology for interpretable models, and more specifically neuron models. In this section we will briefly introduce the key concepts, and explain the correspondence between recurrent neural networks (RNNs) and discretized dynamical systems. We then explain how automatic differentiation has allowed for much more flexibility in defining RNNs. This sets the stage for Chapter 4 (p. 63), where we effectively replace an RNN by a neuron model to infer its parameters.

2.3.1. Deep neural networks

These days of course, familiarity with the basic concepts of neural networks has almost become the norm more than the exception in computational fields, and one can find introductions to the ideas in just about any style or depth. (See for example [1], [53, §5] or [46, §28].) In the following we present the ideas from a physics perspective, and focus on the correspondence to discretized dynamical systems.

Often in physics we want to solve equations that have no closed form solutions. We usually do this by looking for series solutions; typical examples include

- a Taylor series approximation around the minimum of a potential: $V(x) = \sum_n \frac{d^n V(x_0)}{dx^n}(x - x_0)^n$;
- a series of Legendre polynomials describing the zenithal component of spherical harmonics: $\Theta(\theta) = \sum_l a_l P_l$;
- a series of Lagrange polynomials approximating a curve: $y(x) \approx \sum_{j=1}^{k} y_j \, l_j(x)$.

The point we want to make here is simply that even when a closed form solution is unavailable, we can often get very far by writing the true solution as the limit of a sequence of simpler functions.

In the analytical world, we have good tools for differentiating and solving polynomials. So we represent arbitrary functions as sequences of polynomials with unknown coefficients: derivatives can be carried out by hand, and with a bit of ingenuity we can often solve for those coefficients recursively.

Polynomials are not the only way to do this. For example, an integral operator can be represented as the limit of a sequence of increasingly large matrices. [64]

In the computational world, solving polynomial equations is not so easy, because displaying "a bit of ingenuity" is not something computers are very good at. However they are good at other things, including repetitive arithmetic and recursively applying rules. And so for a computer, a more natural thing is to do is to approximate functions not as sequences of polynomials, but as sequences of *compositions*:

$$f(x) \approx \tilde{f}_\theta(x) := (f_\theta^{(d)} \circ \cdots \circ f_\theta^{(1)})(x)\,, \tag{2.17}$$

In Equation (2.17), superscripts indicate function indices, not derivatives.

where $\tilde{f}$ is our approximation of f, and θ is a set of function parameters, analogous to the coefficients of a polynomial expansion. The number d of compositions is often referred to as the *depth*.

Feedforward vs recurrent networks

In a *feedforward* neural network (FNN), each $f_\theta^{(n)}$ in Equation (2.17) depends on a different subset of θ. Each $f_\theta^{(n)}$ is commonly called a *layer*.
In a *recurrent* neural network (RNN), there is typically only one layer, but the network is applied recursively to a stream of inputs. If $x^{(k)}$ and $z^{(k)}$ are the input and output at time step k, then the general form is

$$z^{(k+1)} = f_\theta(x^{(k+1)}, z^{(k)})\,. \tag{2.18}$$

Notice that if there are no inputs $x^{(k)}$ (only an initial value $z^{(0)}$), and we "unroll" the updates for K steps, we end up with a form very similar to Equation (2.17):

$$z^{(k+K)} = (\underbrace{f_\theta \circ \cdots \circ f_\theta}_{k \text{ times}})(z^{(k)})\,. \tag{2.19}$$

The principal difference is that here each "layer" has the same parameters. Notice also that Equation (2.18) is precisely how one would write the update step of any explicit (i.e. forward) numerical scheme for integrating a driven dynamical systems. (With x here playing the role of the independent variable.)
Similarly, Equation (2.19) is how we might write the result of applying this rule to an autonomous system to integrate forward K units of time.
In other words, **a generic dynamical system, when integrated numerically, is equivalent to an RNN**. The consequence of this is that the methods we use to train RNNs will also work for dynamical systems—we just need to first discretize the dynamical system by applying an explicit (i.e. forward) integration scheme.

The basic idea with DNNs is therefore to take a simple, easy-to-compute function $f^{(n)}$, and compose it with itself multiple times.[14] The more often we do that – the "deeper" we make the approximation – the more *expressive* we make $\tilde{f}$. Indeed, as long as the $f^{(n)}$ are nonlinear, we can approximate any function this way; such a result is called a *universal approximation theorem*.[15]

However, showing that a suitable set of parameter values θ *exists* is not the same as actually finding those values. For that we turn to another task that is uniquely well-suited to computers: gradient-based optimization.

14: It doesn't have to be the same function, but this is the simplest and most common approach.

15: The result for deep approximations is surprisingly recent (see Kidger and Lyons [65]), although it's been assumed to be true for decades. This may be because already in 1989, Cybenko [66] showed that infinitely *wide* neural networks have this universal approximation guarantee, and most machine learning researchers felt that was enough to justify their methods. His argument is actually for all nonlinear functions *except* polynomials, but since polynomials are dense in the space of functions, this is mostly a technicality.

2.3.2. Training with gradient descent

As we saw in Section 2.1.3 (p. 33), models are usually fitted by minimizing an objective function. Historically this would have been done with optimization methods like *conjugate gradients*, *BFGS* and *Levenberg Marquardt*, which either rely on convexity or must compute their updates on the entire dataset at every step [67]. In this thesis however we are interested in fitting complex models with analytically intractable[16] and highly non-convex likelihoods, which precludes these classic methods. We are also specifically interested in fitting them to time series data, which means that datasets are relatively large and redundant.[17] As it turns out, training neural networks faces much of the same challenges, and they provide us with the tools we need to learn our models.

By far what has emerged as the most robust way to train neural networks is gradient descent. The idea is deceptively simple: keep going in the opposite direction to the gradient of the loss, and eventually you will hit a local minimum. In other words, if $\dot{\theta}$ is the derivative of the parameters θ with respect to "training time", then training the model amounts to integrating the dynamical system

$$\dot{\theta} \;= -\nabla_\theta\Big(\frac{1}{L}\sum_{x_i\in\mathcal{D}} Q\big(\tilde{f}_\theta(x_i), f(x_i)\big)\Big) \tag{2.20}$$

$$\theta^* \;:= \theta(0) + \int_0^\infty \dot{\theta}(t)\,dt\,. \tag{2.21}$$

In practice the integration is done in discrete steps, with a *learning rate* λ:

$$\theta(k+1) \approx \theta(k) + \lambda\dot{\theta}(k)\,. \tag{2.22}$$

Optimization methods that make use of the gradient aren't new, but experience has shown them to be especially scalable to large datasets. In particular, they are trivially adjusted to use only a small *batch* of data points at every update step:

$$\hat{\dot{\theta}} = -\nabla_\theta\Big(\frac{1}{S}\sum_{\{x_1,\dots,x_S\}\sim\mathcal{D}} Q\big(\tilde{f}_\theta(x_i), f(x_i)\big)\Big)\,. \tag{2.23}$$

This method, called *stochastic gradient descent* (SGD), makes the cost of computing an update step independent of the dataset size L, which is key to enable training on very large datasets.

The main challenge with using gradient descent is that they need to be computed analytically—finite difference methods are simply not accurate enough in high dimensions. This is chiefly enabled by two techniques: *backpropagation* and *automatic differentiation*.

2.3.2.1. Backpropagation

Recall that a deep feedforward neural network is a sequence of compositions trained to approximate a function $f(x)$ (reproduced from Equation (2.17))

$$f(x) \approx \tilde{f}_\theta(x) \;:= (f_\theta^{(d)} \circ \cdots \circ f_\theta^{(1)})(x)\,, \tag{2.24}$$

16: Common examples of optimization problems that *are* analytically tractable include ordinary least squares (see e.g. Chap 7 of [46]), as well as many standard calculations in statistical physics involving Lagrange multipliers.

17: Since it is not atypical for sensors to record at 1 kHz, time series datasets can easily contain hundreds of thousands of data points. They are however temporally correlated, since the process is ostensibly continuous, so information provided by nearby time points is partially redundant.

Note

As one might imagine, Equation (2.22) is not the most accurate way to approximate integrating against the gradient. There is a large body of literature on the best discrete integration schemes, which make use of both variable learning rates and different estimators of the gradient. For most applications, momentum-based methods, and in particular variations of the Adam scheme [68], seem to have established themselves as the best general-purpose methods for these kinds of models.

where θ is a vector of parameters that may be shared between layers. To get the gradient of $Q(f(x), \tilde{f}_\theta(x))$ with respect to θ_i, we apply the chain rule:

$$\frac{dQ}{d\theta_i} = \sum_j \frac{\partial Q}{\partial \tilde{f}_j} \sum_{l_d} \frac{\partial \tilde{f}_j}{\partial f_{l_d}^{(d)}} \left[\frac{\partial f_{l_d}^{(d)}}{\partial \theta_i} + \sum_{l_{d-1}} \frac{\partial f_{l_d}^{(d)}}{\partial f_{l_{d-1}}^{(d-1)}} \left[\cdots \left[\frac{\partial f_{l_d}^{(2)}}{\partial \theta_i} + \sum_{l_1} \frac{\partial f_{l_1}^{(2)}}{\partial f_{l_1}^{(1)}} \frac{d f_{l_1}^{(1)}}{d\theta_i} \right] \cdots \right] \right]. \tag{2.25}$$

A naive implementation of this expression might evaluate it separately for each θ_i, which would end up recomputing the terms $\frac{\partial f_{l_k}^{(k)}}{\partial f_{l_{k-1}}^{(k-1)}}$ many, many times. The idea of the *backprogation* algorithm [69, 46, §16.5.4] is to avoid this by first recursively computing the partial derivatives

$$\begin{gathered} \underbrace{\sum_j \frac{\partial Q}{\partial \tilde{f}_j} \sum_{l_d} \frac{\partial \tilde{f}_j}{\partial f_{l_d}^{(d)}}}_{\overbrace{\sum_j \frac{\partial Q}{\partial \tilde{f}_j} \sum_{l_d} \frac{\partial \tilde{f}_j}{\partial f_{l_d}^{(d)}} \sum_{l_{d-1}} \frac{\partial f_{l_d}^{(d)}}{\partial f_{l_{d-1}}^{(d-1)}}}^{\overbrace{\sum_j \frac{\partial Q}{\partial \tilde{f}_j} \sum_{l_d} \frac{\partial \tilde{f}_j}{\partial f_{l_d}^{(d)}} \sum_{l_{d-1}} \frac{\partial f_{l_d}^{(d)}}{\partial f_{l_{d-1}}^{(d-1)}} \sum_{l_{d-2}} \frac{\partial f_{l_{d-1}}^{(d-1)}}{\partial f_{l_{d-2}}^{(d-2)}}}^{\cdots}}} \\ \cdots \end{gathered} \tag{2.26}$$

and then multiplying them by $\frac{\partial f_{l_k}^{(k)}}{\partial \theta_i}$ to get the contribution to the gradient from each layer.

The braces in Equation (2.26) represent the error signal being "propagated backwards" from the output layer to the input layer.

The backpropagation algorithm makes the computational cost of computing gradients essentially linear in the number of layers, and thus enables learning with deep neural networks.

Training *recurrent* neural networks (and therefore also discretized dynamical systems) uses a simple generalization of this approach: we unroll K updates to obtain an expression like Equation (2.19) (p. 39), then apply backpropagation as with the feedforward network. The unroll depth K is a training parameter; ideally it is chosen large enough that the updates K steps in the past don't meaningfully contribute to the gradient.

This strategy of computing gradients on an unrolled RNN is appropriately named **backpropagation through time (BPTT)** [1, §10].

Since modern software packages providing automatic differentiation are typically designed for machine learning applications, their gradient function is usually already implemented to exploit backpropagation.[18]

18: PyTorch even calls its gradient function `backward`.

2.3.3. Automatic differentiation

Until the 2010's, machine learning researchers would spend a lot of time manually calculating (and then correctly implementing) gradients for their

models [70, §18.4.2]. Model design was limited by the need for analytically tractable gradients.

It is understandable therefore that the introduction of automatic differentiation (autodiff) tools, which could reliably differentiate almost arbitrarily deep networks in seconds, was something of a revolution for the field.[19] Most relevant for our purposes, by allowing to train networks where $\tilde{f}$ is almost arbitrary, this technology allows us to move beyond standard neural network architectures, and replace them with full-blown dynamical systems. In fact, the work in this thesis is almost entirely dependent, in one way or another, on techniques made possible by automatic differentiation.

19: There are now many libraries that perform automatic differentiation. Python likely has the most, with *Theano* (now *PyTensor*), *Autograd*, *TensorFlow*, *PyTorch*, and *JAX*, but there are also e.g. packages for Julia (*JuliaDiff*) and Lua (*Torch*).

What makes manually computing gradients hard is the need to recursively apply the chain rule. On the other hand, recursively applying rules is something computers are very good at, so it makes sense to automate the procedure. Conceptually it is also not particularly difficult: we just need

- a dictionary of differentiation rules for elementary functions,
- a function that applies them recursively,
- and a format for representing mathematical expression.

Listing 2.1 is a simple Python program implementing the first two elements, and using Polish notation to represent mathematical expressions.

```
diffrules = {
  "+"   : lambda a,b: ("+", diff(a), diff(b)),
  "*"   : lambda a,b: ("+", ("*", diff(a), b),
                            ("*", a, diff(b))),
  "^"   : lambda a,n: ("*", ("*", n, ("^", a, ("+",n,-1))),
                            diff(a)),
  "sin": lambda x   : ("*", ("cos", x), diff(x)),
  "cos": lambda x   : ("*", ("*", -1, ("cos", x)),
                            diff(x)),
  "exp": lambda x   : ("*", ("exp", x), diff(x))
}

def diff(expr):
    global wrt
    if isinstance(expr, tuple):
        return diffrules[expr[0]](*expr[1:])
    elif expr == wrt:
        return 1
    else:
        return 0

expr = ("+", ("sin", ("^", "x", "n")),
             ("exp", ("*", 2, "x")))
wrt = "x"  # Set the variable wrt which we differentiate
diff(expr)
```

Listing 2.1: A simple automatic-differentiation program
`diffrules`: defines analytic derivatives for addition (+), multiplication (*), exponentiation (^), sin, cos and exp.
`diff()`: recursively applies these rules.
`wrt` is the variable **w**ith **r**espect **t**o which we are differentiating.

Expressions are given using Polish notation; e.g. the test value is `expr` = $\sin(x^n) + \exp(2x)$.
After executing this code, one finds `diff(expr)` = $\cos(x^n)nx^{n-1} + 2e^{2x}$, as expected.

As always however, the devil is in the details. For example, our simplistic implementation would incorrectly differentiate x^0 as x^{-1}. And of course specifying expressions using Polish notation quickly grows into a disorienting heap of parentheses. So most of the work in making an actually useful autodiff tool goes into corner cases and improving the user interface.

Some autodiff libraries use an approach similar to Listing 2.1, in that expressions are evaluated immediately; this is the case for example of *PyTorch*, and *TensorFlow* in eager mode. These libraries use caching to avoid recomputing expressions, and thereby benefit from the aforementioned (p. 40) advantages of backpropagation, but otherwise behave much like standard imperative code. This makes it possible to use control flow statements like `if` and `for` loops without issue.

Other libraries on the other hand will first build an *expression graph* for the entire expression being differentiated, like the one shown in Figure 2.2. *Theano* [71] (and its descendant *PyTensor*) most fully embrace this approach. Because the graph is static (it is fully constructed before computations start), this approach is more restricted in its support for control flow statements.

What one gains with expression graphs however is the ability to inspect and freely modify them. This allows expressions to be constructed non-sequentially, by placing placeholder variables that are later replaced by complete expressions. The software developed for the work in Chapter 4 (p. 63) uses this ability extensively, in order to construct a full update step from multiple inter-dependent equations.

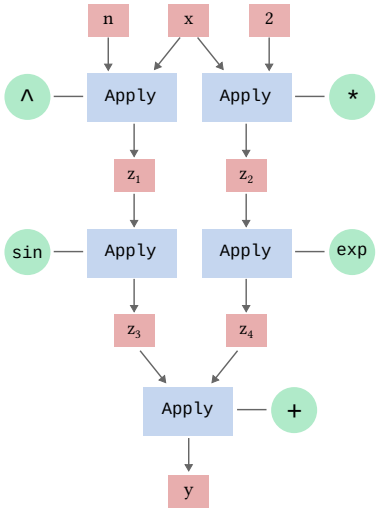


Figure 2.2.: Expression graph for $\sin(x^n) + \exp(2x)$. Representation of the `expr` $\sin(x^n) + \exp(2x)$ from Listing 2.1 as an *expression graph*. This is the form used by Theano: the `Apply` nodes are responsible for most of the logic.

Statistical theory of model selection

3.

3.1. Model comparison as a foundation of induction

In many ways, the scientific method can be viewed as a process for performing *induction*: inferring *general* laws of nature from the results of *particular* experiments. Karl Popper suggested to anchor the logical foundations of this approach on the idea of *falsification*; by seeking out predictions that might be disproven by experiments, scientists minimize confirmation biases when testing their theories. According to Popper, a theory enjoys scientific acceptance until disproven, at which point it is *falsified*, and must be replaced by a better one. Theories thus improve monotonically over time.

An important limitation of the Popperian framework is that, while simple and compelling, it is at odds with how most science progresses in reality. As later noted by Kuhn [72], we usually don't immediately reject a theory when it fails to predict experiments—after all, an imperfect theory is still better than no theory at all. Instead, we reject a theory only when a *better* one is available. Indeed, most theoretical models (and in particular all the neuron models we described in Chapter 1 (p. 6)) are known to be approximate, and therefore "falsifiable" *by design*. So while the practice of designing experiments with a goal to infirm (rather than confirm) remains essential, the ultimate rejection of a theory is based on *comparison* rather than falsificaton. Sixty years on, this continues in many cases to be an accurate reflection of the scientific process, and will be the point of view we take in Chapter 5 (p. 99).[1]

In many ways, the process of learning a model we described in Chapter 2 (p. 29) is analogous to developing a theory: we start by making informed *assumptions* about the system of interest (such as the way specific variables interact, or particular symmetries those interactions should satisfy), then—based on observed data—*deduce* (or "learn") what this implies about the model's parameters. Finally we *test* that the model generalizes to held-out data.

Nevertheless, to frame the problem of model selection, it can be useful to think of "theory" and "model" as expressing different ontological expectations. We expect a **theory** to reflect some kind of truth about the world, and therefore to be perfectly reproducible. For example, the properties of fermions are unchanged whether we probe them in Ottawa or Tokyo—any measured differences are ascribed to imperfections of the experiment. This situation is at least ontologically compatible with the Popperian perspective, which also supposes the existence of a true model.[2] This makes induction relatively straightforward: the aim is to recover the unique true model, which by virtue of its uniqueness will generalize to new experiments. And since the true model should predict observations perfectly, if we have sufficiently minimized experimental errors, it will also be the one with the lowest training loss.

1: There has of course been plenty of work on the scientific method since Kuhn, but for our purposes his perspective suffices.

2: Much of frequentist statistics also assumes the existence of a true model. The relation goes both ways: many modern attempts at falsifying a theory, in particular in physics and astronomy, take the form of a hypothesis test (where rejecting the null hypothesis is equated with falsifying the theory).

In contrast, we will consider that a **model** is used to describe systems that can only be reproduced approximately, even in an idealized, perfectly controlled experiment: there may be strong statistical similarities, but no universal true model is preserved across experiments. This is the case of most biological experiments, since (in contrast to atomic particles) no two cells or individuals are perfectly identical. Since in this thesis we focus on models for neuroscience, we will always be squarely in the realm of models. Imperfect reproduction is far from limited to biological sciences however: even Newton's laws become imperfect at the quantum scale, and often effective models, by virtue of their greater simplicity, can better support intuition than an exact theory.

The absence of a universal true model however poses a new problem from the point of view of induction: the approximations (and therefore the model) that work best for one experiment may not generalize to another experiment. So it is not enough to simply select the one with the lowest loss. We need to also guard against the risk of *overfitting* to the data.

Note that due to experimental uncertainty, overfitting is also a concern when learning a theory: an incorrect model can still have a lower loss, causing it to be selected in favour of the true model.

We devote this chapter to a survey of existing methods for comparing models, and contrast how they address overfitting. This is the context against which the methods presented in Chapter 5 (p. 99) are developed.

Notation

Compared to the rest of the thesis, this chapter has a much stronger statistical focus. For this reason we use somewhat different notation, which is not only better suited to the topic, but also closer to what one finds in the corresponding literature. The notation is specifically meant to simplify tracking the size of the dataset (which in this chapter we denote n) and what data are used to obtain an estimate (denoted by a subscript $_{|y^n}$).

- $y_i \in \mathbb{R}^k$: individual data sample
- $y_{i,l}$: component l of sample i
- y^n, $\mathcal{D}$: *observed* dataset containing n samples
- $\tilde{y}^n$: *unobserved* dataset containing n samples
- y_l^n: dataset obtained by taking the l-th component of each sample
- Often predictive models will have an independent variable x and a dependent variable y, with the model itself written $p(y \mid x)$.
  - Presumably the independent variables x are themselves sampled with some probability $p(x)$. Then all the formalisms are exactly the same, as long as we take samples to be not just the y_i, but the tuples (x_i, y_i), with probability $p(x, y) = p(y \mid x)\, p(x)$.
  - Datasets can then be written (x^n, y^n), $(\tilde{x}^n, \tilde{y}^n)$, etc.
- $\mathcal{M}$: a probabilistic *model* (i.e. a distribution), either of the form $p(y \mid \theta)$ or $p(y \mid x; \theta)$.
  - A model is the same as a *hypothesis class* (see Section 2.1.1 (p. 31))
- θ, ν: vector of distribution parameters.
- θ_0: vector of true distribution parameters, assuming it exists.
  - Note: when we write out the parameter vector, e.g. $\theta_0 = (\mu, \sigma)$, we usually omit the subscript.
- $\hat{\theta}$, $\hat{\mu}$: Estimator for parameter θ, μ. Estimators are always obtained for some observed dataset, and we may make this explicit by writing e.g. $\hat{\mu}_{|y^n}$.
  - If these are obtained by maximizing the likelihood, we call this the *maximum likelihood estimate* (MLE).
- *loss*, ℓ: A functional used to compare models based on their predictive performance
  - Note that although it depends on the data, the loss is formally a function of the *parameters*. (This is a key conceptual difference between a likelihood and a conditional probability: the latter is a function of the data that depends on the parameters.)
- $p(y)$ or $p(x, y)$: True data distribution.
- $q_\theta(y)$ or $q_\theta(x, y)$: Approximation to p, parameterized by θ.
- $\tilde{q}_\psi$ or $\tilde{q}_{\tilde{\psi}}$: A different approximation to p, parameterized by a different set of parameters ψ.
- $q_{\theta|y^n}$: the approximation obtained by fitting q to the data y^n.

Assumption of i.i.d. data

In this chapter we always assume that datasets are composed of i.i.d. samples. The most common situations where data are *not* i.i.d. are time series data; in this case, results are easily extended by either thinning the dataset, or defining a notion of "effective sample size".

Scoring rule ≠ score function

The term "scoring rule" is also used to refer to the functional used to compare models. This is very different from the "score function", which statisticians use to refer to the derivative of the log likelihood. To avoid confusion, we exclusively use the terms "loss" and "criterion".

3.2. Avoiding overfitting in data-limited scenarios

In Section 2.1.4 (p. 34), we saw that it is *de rigueur* with overparameterized models to split a dataset into train and test subsets in order to avoid overfitting. In data-limited scenarios however, this may not be satisfactory: we want the absolute best possible model, and reserving say 20% of the data for testing might increase the variance on the empirical risk $\hat{R}(\theta)$ (p. 32) so much that we no longer select the correct model.

Classic statistics usually presumes that models are compared in this data-limited regime: the data may for example be obtained by running expensive and time-consuming surveys, and we want to make the most of each sample. In this case we still have a few options to estimate the loss and (hopefully) select the best model.

Generalization criteria (p. 49) compare models based on their *expected loss* on *new data*.

This includes the method of splitting the dataset into training and test sets from Section 2.1.4 (p. 34), but also *cross-validation*, which improves the accuracy of the estimated loss at the cost of longer computation time.

Replication criteria compare models based on their *expected loss* on data from a *new data-generating process*.

In machine learning, this can also be referred to as a model's ability to generalize "out-of-distribution". It is related to *transfer learning* and can be estimated by testing models on new datasets that were not used for training. Results are of course always dependent on the choice of that external dataset.

In science, this would correspond to *replicating* an experiment in a new setting or with a new subject.

In all cases, there must be some form of similarity between the training dataset and the test dataset.

Since the new data-generating process is by definition undefined, estimating the loss under replication is necessarily an ill-posed problem. This may be why it is largely ignored in the statistical literature.

Nevertheless, since replication is the basis for scientific induction, this is ideally how we would compare learned models. **In Chapter 5 (p. 99) we develop a comparison theory that can account for variations across replications.**

Complexity criteria (p. 51) seek to minimize a measure of *complexity* of the model. Since they are used to select the model with lowest (i.e. simplest) *complexity*, they are often explained as implementing a version of Occam's razor.

One gets the impression that most applied statisticians working with complex models consider cross-validation the clearly superior approach, likely for the same reasons that machine learning practitioners will always use a train/test split (p. 34): it does not require making assumptions on the true data-generating process to extrapolate beyond the observed data (instead using the data themselves), and is thus much more robust to misspecification. Other types of criteria however still have many proponents, especially in theoretical work.

Information criteria (p. 56) compare models based on their *likelihood* on the *observed data*. A correction term is required, in order to correct for the overfitting due to using the same data to both train and test the model, resulting in the typical "loss + correction" form.

Precisely estimating the overfitting bias term by information criteria is essentially an impossible task, since it depends both on the true data distribution and on the specifics of each model. Approximations are therefore inevitable. The different choices of approximations are what distinguish different information criteria.

3.2.1. Overview of common selection criteria

At the most basic level, model comparison involves defining a functional on models—typically called in this context a *criterion*—and selecting the one that returns the lowest value. What makes a criterion different from the loss functional used for training is the need to account for overfitting; the kind of overfitting we wish to account for largely determines the form of the criterion.

In Table 3.1 we list some of the most common comparison criteria, which will be described in later sections. Each criterion follows from different assumptions about the problem being solved, and therefore the nature of overfitting; to make these assumptions comparable, we reduce these assumptions to few binary descriptors:

Statistical framework Most criteria are developed within a frequentist or Bayesian framework, although there are exceptions: the "luckiness" functions used by minimum description length (MDL; Section 3.4.4 (p. 54)) are not entirely equivalent to Bayesian priors, while our own EMD approach (described in Chapter 5 (p. 99)) is based on empirical risk minimization ERM (Section 2.1.2 (p. 31)).

Model form A criterion applied to *structural models* compares two families of models, each defined by a set of equations and spanned by their possible parameter values. I.e. it takes the form $B(\mathcal{M}_A) \overset{?}{<} B(\mathcal{M}_B)$.

This differs from the machine learning methods described in Chapter 2 (p. 29), where performance is always measured for specific values of those *parameters*; the associated comparison would take the form $B(\theta_A) \overset{?}{<} B(\theta_B)$.

Stationary data-generating process Whether we assume that experiments will always generate samples from exactly the same distributions This is equivalent to requiring that experiments, *including their distribution of measurement errors*, are always exactly replicated.

When the data-generating process is stationary, one can unambiguously define it to be the "true model".

Misspecified models We say that a model is *misspecified* if it does not contain the true data-generating process in its hypothesis class. (The notion of misspecification usually presumes the existence of a true model.)

Many criteria require that one of the model candidates be well-specified, i.e. that the true model exists and is contained in one of the candidate models.

Do we account for performance on new data? "Yes" indicates that we are interested in a model's predictive performance. In machine learning, this is

Note

A criterion that assumes experiments to be perfectly reproducible will not consider the performance on "out-of-distribution" samples, since it assumes that samples are always drawn from the same distribution as the training data.

Terminology: Consistency

A model comparison criterion is *consistent* if, when the true model is among those compared, it is recovered in the limit of infinite data. Note that this is slightly different from the notion of consistency for an estimator, which we defined in Section 2.1.2 (p. 31) (see esp. Figure 2.1): for an estimator, consistency does not require the existence of a true model, only that the estimator converge to its infinimum.

Table 3.1.: Descriptors for some common model selection methods. (MDL: minimum description length. MST: model selection test [74]. ERM: empirical risk minimization (Section 2.1.2 (p. 31)). EMD: empirical modelling discrepancy (Chapter 5 (p. 99)).) MST and EMD are not part of the standard statistics corpus, but noteworthy for making different modelling assumptions.

		Model assumptions				Scoring considerations		
Method	Statistical framework	Model definition	Stationary data-generating process?	Models may be misspecified	Models may be singular	Generalization error? (in-dist)	Replication error? (out-of-dist)	Simplicity bias?
Bayes factor	Bayesian	structure	**yes**	no	no	no	no	**yes**
WAIC	Bayesian	structure	**yes**	**yes**	**yes**	**yes**	no	no
AIC / likelihood ratio	Frequentist	structure	**yes**	no	no	**yes**	no	no
BIC	Bayesian	structure	**yes**	no	no	no	no	**yes**
DIC	Bayesian	structure	**yes**	no	no	no	no	**yes**
MDL	MDL	structure	**yes**	no	no	**yes**	no	**yes**
MST	Frequentist	structure	**yes**	**yes**	no	**yes**	no	no
EMD	ERM	structure + parameters	no	**yes**	**yes**	**yes**	**yes**	no

achieved by having separate datasets for training and testing (see Section 2.1.4 (p. 34)).

"No" indicates that we only care about the performance on data we already have. This frame only makes sense if we also expect to learn a true model, otherwise the learned model would be anecdotal.

Performance may be tested on samples generated from exactly the same distribution as the original experiment ("in-distribution"), or from a different distribution ("out-of-distribution").

Models may be singular A model is *singular* if for some parameter values, its Fisher information matrix is singular (i.e. has one or more vanishing eigenvalues).

Intuitively, a model is singular if not all of its parameters are identifiable [73].

Neural networks, being extremely overparametrized, are perhaps the most salient examples of singular models, but most interpretable models—including those we are interested in this thesis—are singular [73].

Criteria derived using Wilks' theorem (see Section 3.5.2 (p. 59)) are invalid with singular models.

Do we apply a bias towards simpler models? Whether the criterion incorporates a measure of model complexity, so that given two models with the same accuracy, it will prefer the simpler one.

We also include in Table 3.1 the model selection tests of Golden [74], which, while not really widespread in the literature, stand out for not requiring that one of the model candidates be the true data-generating process. The last line (empirical modelling discrepancy) is our own method, which will be presented in Chapter 5 (p. 99).

Table 3.1 highlights one of the conceptual differences between machine learn-

Note

Describing "out-of-distribution" performance is necessarily ill-posed, since there is no unique way to define such distributions.

Note

The descriptors in Table 3.1 indicate reproducibility features a method attempts to account for. This is done by making assumptions, which may introduce further limitations on a criterion's validity.

ing and much of theoretical statistics: whereas out-of-distribution performance is a key consideration for machine learning models (see Section 2.1.4 (p. 34)), since it is ill-posed, it is largely ignored in the statistics literature.

Comment: Concentration of the likelihood

When considering how information criteria work in practice, it is important to keep in mind that the likelihood has a strong tendency to concentrate as we add data samples. E.g. when the samples are independent, we have

$$q(y^n|\theta) = \prod_{i=1}^{n} q(y_i|\theta) \quad \text{and} \quad \log q(y^n|\theta) = \sum_{i=1}^{n} \log q(y_i|\theta)\,. \tag{3.1}$$

In other words, already for moderate n, $q(y^n|\theta)$ is a very peaked distribution, and it quickly approaches a Dirac delta as n is increased. This "concentration of the measure" allows one to use Laplace's approximation, which is convenient for theoretical proofs, but it can make fits more vulnerable to outliers or misspecification errors.

3.3. Generalization criteria

Generalization criteria compare models based on their *generalization error*, i.e. the expected loss on new samples:

$$\mathbb{E}_{\tilde{y}\sim p}[\text{loss}]\,. \tag{3.2}$$

In the context of *learning machines* [75], which we considered in Chapter 2 (p. 29), we usually have enough data that reserving say 20% of the samples for the test set does not meaningfully change the trained model. The same can be said for many experimental contexts, where it is usually possible to repeat an experiment until sufficient data have been gathered.

Sometimes however, data collection is more limited. It is in this context that statistics has developed methods to both train a model and estimate its generalization error (Equation (3.2)) that make more efficient use of the available data.

Important

The loss in Equation (3.2) can be any scalar functional of the data, which makes generalization criteria more flexible than complexity criteria (which use a fixed expression for the model complexity) or information criteria (which always compare the expectation of the log likelihood). This allows for a more operational criterion, since one can tailor the loss functional to the application.
Note also that the expectation in Equation (3.2) is over individual samples y. In contrast, both complexity and information criteria estimate an expectation over the entire datasets y^n, and are thus sensitive to the size of that dataset.

3.3.1. Cross-validation

The first of these methods is *cross-validation*. The idea is fairly simple: instead of a single train/test split, use multiple, so that in the end every sample has served to test the model once. For example, an 80%–20% split would typically result in 5 validation sets. A more conservative split, like 90%–10%, would result in more accurate estimates of the loss, but require more time (since the model needs to be retrained 10 times instead of 5).

Note

Since the same sample is never used for both training and testing, the generalization error computed from each testing subset is an unbiased (albeit possibly very noisy) estimate of the true generalization error (Equation (3.2)). Averaging across validation splits reduces the noise on that estimate.

To obtain the final estimate for the generalization error, one then averages the generalization errors for each train/test split. In low-data scenarios, this can substantially improve the accuracy of the generalization error—at the cost of increased computation time.

Once a model is selected, it would then be retrained using the entire dataset, thus obtaining the most accurate possible model.[3]

3: Again, this is relevant because we are in a data-constrained scenario, where 20% more training data can meaningfully improve a model's accuracy.

3.3.2. Leave-one-out cross-validation (LOO-CV)

Leave-one-out cross-validation (LOO-CV) is the most conservative (and therefore also the most expensive) form of cross-validation, where the test sets consist of just one sample: a dataset with n samples is retrained n times, each time with $n-1$ samples. This is often considered the gold standard in terms of estimating generalization error [76].

However in the context of Bayesian models or learning machines, in which training each model training can be very expensive, LOO-CV is also often computationally impractable.

3.3.3. Widely applicable information criterion (WAIC)

Watanabe's WAIC criterion is an approximation of the generalization error (Equation (3.2)) that has been shown to be asymptotically equivalent to LOO-CV [77]. In contrast to other comparison criteria, its approximations continue to hold for learning machines and singular models.[4] It can thus be viewed as a computationally tractable approximation to LOO-CV.

4: Recall that a model is *singular* if some of its parameters are non-identifiable.

Note

In contrast to information criteria, any loss functional can be used to compare models with the WAIC—not just a log probability [56, p. 2]. Nevertheless, we restrict our exposition to the *log posterior*, both for concreteness and to simplify analogies with information criteria.

Asymptotic criterion

Like other criteria, the WAIC is derived in the asymptotic limit, and so requires a large n.

The key result underpinning the WAIC is a set of equations of state relating both generalization and training errors [55, §3.1]. After further characterizing the difference between two types of training errors [55, §5], Watanabe ultimately proposes the WAIC criterion. The form that has become standard is

$$\mathrm{WAIC}(q; y^n) := 2 \sum_{i=1}^{n} \log\Big[\mathbb{E}_{\theta|y^n}\big[q(y^n \mid \theta)\big]\Big] - 2\,\hat{p}_{\mathrm{WAIC}}\,, \tag{3.3}$$

where [56, Eq. 13]

$$\hat{p}_{\mathrm{WAIC}} = \sum_{i=1}^{n} \mathrm{Var}_{\theta|y^n}\big[\log p(y_i|\theta)\big]\,. \tag{3.4}$$

can be interpreted is a measure of the parameter space's complexity.

The factor two is included to match the form of *information criteria*, introduced below (p. 56). Note however that although the WAIC shares the *name* of information criteria, and reproduces their functional form, it **estimates a different quantity** (generalization error vs. overfitting bias for n samples) **and is obtained via a completely different argument**, with different (and esp. weaker) assumptions on the models.

Hint

Compared to simple information criteria, computing the WAIC is still relatively expensive, since it remains a Bayesian quantity and thus requires a large number of i.i.d. samples drawn from the Bayesian posterior (typically generated with MCMC as part of Bayesian inference, which we described in Section 2.2 (p. 35)). Given those samples however, computing the WAIC is straightforward, since general purpose functions are available—for instance, the ArviZ project [78] maintains implementations for Python and Julia.
Good implementations of the WAIC will also make use of importance weighting techniques [56], which substantially improve the robustness of the estimated generalization error compared to naive implementations.

3.4. Complexity criteria

3.4.1. Parameter complexity — Bayesian model evidence

Given a prior π over the parameters, the **posterior density** in Bayesian statistics is defined as (reproduced from Equation (2.13) (p. 36))

$$p_{\mathrm{post}}(\theta \mid y^n; \mathcal{M}) = \frac{p(y^n \mid \theta; \mathcal{M})\,\pi(\theta)}{p(y^n \mid \mathcal{M})} \qquad \text{(posterior)}. \tag{3.5}$$

Since the denominator is independent of the parameters, it contributes only a constant to the log posterior: it does not change its shape, and thus is usually

ignored in Bayesian *inference*. In particular, the **maximum a posteriori (MAP)**—defined as

$$\theta_{\text{MAP}} = \operatorname{argmax} P(\theta \mid y^n; \mathcal{M}) \tag{3.6}$$

—can be computed by maximizing only the numerator of Equation (3.5).

Recall that classic statistics frames model selection as one of comparing different *structural models*, rather than specific parameter values.

In Bayesian *model comparison* however, we specifically want a quantity that depends only on the model *structure*, with no dependence on any specific parameters. The denominator—which is referred to as the **model evidence**—is a natural way to do this. Since the posterior probability is normalized, the model evidence can be computed by marginalizing the likelihood over the prior:

$$p(y^n \mid \mathcal{M}) = \int p(y^n \mid \theta; \mathcal{M})\, \pi(\theta) d\theta \qquad \text{(evidence)}. \tag{3.7}$$

In this way, the model evidence is akin to a likelihood for only the *structure* of the model, independent any particular parameter values. Methods for computing the evidence tend to be especially reliant on the likelihood having a simple shape, in order to carry out the marginalization in Equation (3.7).

Sensitivity of the evidence on the prior

In contrast to Bayesian inference, which considers only the numerator of Equation (3.5), the model evidence is highly sensitive to the choice of prior. In some cases—in particular if the set of models is parameterized—the sensitivity to the prior may become so strong that the data become irrelevant [79].

Sensitivity of the evidence to dataset size

Due to the propensity of the likelihood to concentrate around its maxima with dataset size, the model evidence also strongly depends on the number of data points n.

Another consequence of the likelihood concentrating around its maximum is that direct integration of Equation (3.7) is often *numerically* intractable. Instead, stochastic methods like *nested sampling* [80, 81] are typically used, which automatically adjust the integration domain to maximize a computation budget.

Alternatively, if the dataset is large enough (and the likelihood is positive definite), the likelihood becomes so concentrated that it can be approximated by a Gaussian—thus making it *analytically* tractable. This is the approach used by the Bayesian information criterion described below.

Note

Nested sampling is how we compute the model evidence in Section 5.2.9 (p. 119).

3.4.2. Bayes factor

The **Bayes factor** between two models is the *ratio of their model evidence*:

$$B_{AB}^{\text{Bayes}} = \frac{p(y^n \mid \mathcal{M}_A)}{p(y^n \mid \mathcal{M}_B)}. \tag{3.8}$$

Traditionally, values for B_{AB}^{Bayes} are interpreted using Jeffrey's scale[5] [82–85],

5: There are a few different variants of Jeffreys' scale; the one we present in Table 3.2 was proposed by Kass and Raftery [82].

B_{AB}^{Bayes}	$2 \ln B_{AB}^{\text{Bayes}}$	Grade of evidence in favour of $\mathcal{M}_A$
1 to 3	0 to 2	Barely worth mentioning
3 to 20	2 to 6	Positive
20 to 150	6 to 10	Strong
>150	>10	Very strong

Table 3.2.: Jeffreys' scale [82], in both ratio and logarithmic grades. For the grades when $\mathcal{M}_B$ is the preferred model, use reciprocals of the first column, or negative values of the second column.

with the different grades of evidence listed in Table 3.2. Note that often the logarithm of B^{Bayes} is easier to work with, both for reasons of symmetry ($\log B_{AB}^{\text{Bayes}} = -\log B_{BA}^{\text{Bayes}}$) and to avoid numerical underflows.

Limited interpretability of Bayes factors

Equation (3.8) is most reasonable when the y take discrete values, so that the numerator and denominator are probability *masses*. If however the y take continuous values, then Equation (3.8) is a ratio of probability *densities*, and its value is not reliable.
Even with discrete values however, the values of the Bayes factor are not entirely reliable, due to the sensitivity of the evidence to the prior [79] and to the number of samples.[a] This may explain why, despite usage going back decades, the evidence supporting Jeffreys' scale remains anecdotal.

[a] Note for example that since likelihoods concentrate at different rates, we can make the ratio in Equation (3.8) arbitrarily big or small simply by recording more data, i.e. by increasing n—without any change to the underlying distributions.

3.4.3. Bayesian information criterion (BIC)

Since the MAP maximizes the posterior w.r.t. to the parameters, the gradient of the posterior density $\nabla_\theta(p_{\text{post}})$ vanishes at θ_{MAP}. We can therefore approximate the posterior in a small neighbourhood around θ_{MAP} with a second-order Taylor approximation. If the posterior is sufficiently concentrated around the MAP that this small neighbourhood contains almost all of its probability mass (as tends to happen when the number of samples n increases), we can approximate it with a Gaussian centered on θ_{MAP}.[6] This is called the *Laplace approximation*, and generally holds when the number of samples n is sufficiently large; sufficient conditions are given by the Bernstein-von Mises theorem.

The *Bayesian information criterion* (BIC), proposed by Schwarz [86], is the application of the Laplace approximation to compute the model evidence. Schwarz [86] derives the form

$$\begin{aligned} \text{BIC}(q; y^n) &= -2 \log q_{|y^n}(y^n|\hat{\theta}_{\text{MAP}}) + k \log(n) \\ &\approx \int q(y^n|\theta)\, \pi(\theta) d\theta \,, \end{aligned} \tag{3.9}$$

where $q_{|y^n}$ is the posterior given the data y^n, k is the number of parameters and n the number of samples. This reproduces the "likelihood + correction" form of information criteria (p. 56), with one key difference: the correction term scales (logarithmically) with the dataset size. This reflects the fact that

6: In essence, the quadratic term of the Taylor approximation becomes the quadratic argument in the Gaussian's p.d.f.

Note

The use of a Taylor approximation mirrors the derivations of *information criteria*, described in Section 3.5 (p. 56) (and especially in Section 3.5.2 (p. 59)). Note however that here we use it to compute the integral of the likelihood over the prior, rather than the underestimation of the loss due to overfitting.

while information criteria compute the underestimation of the loss due to overfitting, the BIC computes a fundamentally different quantity, namely the model evidence. Reflecting this difference, the $k \log(n)$ term is usually interpreted as quantifying the *complexity* of the model, rather than a correction for overfitting.

BIC is inconsistent when models are misspecified

When the problem is misspecified (i.e. none of the models contain the true data-generating process), the complexity term $k \log n$ can become so large that the BIC ignores which model actually fits the data better and simply selects the one with the fewest parameters [88].

BIC is invalid for singular models

The Laplace approximation absolutely requires that the model's Fisher information matrix be positive definite, which is not the case for the learning machines of Chapter 2 or the interpretable models we will study later in this thesis.
The BIC also may not be consistent with high-dimensional models [87, p. 491].

BIC is not an information criterion

Since the BIC estimates a different quantity than information criteria, "it is completely possible for a complicated model to predict well and have a low AIC, DIC, and WAIC, but, because of the penalty function, to have a relatively high (that is, poor) BIC." [79, p. 175]

Note

The original derivation [86] nominally starts from the MAP, but immediately switches to the MLE since in the asymptotic regime they are the same. Conceptually however, the MAP is what best corresponds to what the BIC purports to estimate, and so should be preferred. Usage however is not consistent: e.g. Gelman et al. [76, p. 175] give the definition in terms of the MAP, while Spiegelhalter et al. [87] use the MLE.

3.4.4. Minimum description length

The *minimum description length* (MDL) principle is another way to formalize the idea of picking the simplest model that fits the data, based on ideas of coding theory instead of the Bayesian model evidence. It was originally proposed by Rissanen [89] in 1978, and by 1998 was an established theory [90]. In the early 2000's it was taken up by Grünwald [91], who, along with collaborators, moved the theory away from its coding theory roots in order to apply it to a wider range of problems. In doing so, they discovered that the formalism is in fact practically equivalent to a Bayesian formalism [92], albeit with different interpretations and modelling choices. It is this approach we describe in this section, following especially the presentation of Grünwald and Roos [92].

The basic idea of MDL is to measure the complexity of a model by the length of the *code* required to encode *both* the model *and* the data. The minimal number of bits required to do this is the minimum description length. A key theoretical tool here is the *Kraft inequality*, which states that given a probability distribution p over a countable set Y, one can devise a lossless code that encodes each $y \in Y$ into $-\log p(y)$ bits (rounded up to the nearest integer).[7] A Bayesian model $\mathcal{M}$ with prior π would therefore need at least

Note

Our exposition is based on [92], which is noteworthy for drawing an explicit link between MDL and Bayesian methods. We should note however that this is just one of at least two distinct approaches of MDL, the other being that of Rissanen et al. [89, 90]. Both purport to compute the same quantity (the minimum description length), but use different methods to do so.

For a description of MDL from the point of view of empirical risk minimization theory, see [48].

7: Note that, if we ignore the rounding, the average length of a word under such a code would be $\mathbb{E}_{x\sim p}[\log p(x)]$, i.e. the entropy. Shannon's source coding theorem states that this is the shortest possible average code length.

$$\sum_{\theta\in\Theta} -\log \pi(\theta) + \sum_{i=1}^{n} -\log p(y_i \mid \mathcal{M}_\pi) \tag{3.10}$$

bits to be encoded. (The first term corresponds to encoding the prior, and the second to encoding the data, with help from the prior.[8]) All else being equal, a model with more parameters would require more bits to encode the prior, and thus have higher MDL.

The original approach to MDL centered on devising explicit codes, for which one could minimize the length for each model to obtain its MDL. The more modern approach championed by Grünwald et al. is to simply define the code length as $-\log p$, since this is a tight lower bound.[9]

In some sense, the MDL formalism is intrinsically Bayesian: models are parametrized, and prediction probabilities obtained by multiplying a likelihood and a prior.[10] In fact it goes even further, assigning a "hyperprior" κ on the discrete set of models, so that the code length is actually (c.f. Equation (3.10))

$$\mathrm{MDL}(y^n, \mathcal{M}_\pi) = -\log \kappa(\mathcal{M}_\pi) + \sum_{\theta\in\Theta} -\log \pi(\theta) + \sum_{i=1}^{n} -\log p(y_i \mid \mathcal{M}_\pi)\,. \tag{3.11}$$

As with other criteria, MDL seeks to compare models based on their *structure* rather than any particular parameter values. This is done by defining a *universal distribution* that does not depend on model parameters. The simplest universal distribution is likely the *model evidence* (Equation (3.7) (p. 52)), which involves integrating over the prior. However by far the preferred universal distribution within MDL is the *normalized maximum likelihood (NML)*:

$$p^{\mathrm{NML}}(y^n \mid \mathcal{M}_\pi) := \frac{\max_\theta p(y^n \mid \theta; \mathcal{M})\,\pi(\theta)}{\int \max_\theta p(y^n \mid \theta; \mathcal{M})\,\pi(\theta) dy^n}\,. \tag{3.12}$$

The numerator in Equation (3.12) is just the maximum a posteriori fit; here it is $\max_\theta$ which "gets rid of" the dependence on the parameters θ. This will of course tend to overfit to the data, especially if the model is very flexible. However, notice how the normalization constant in the denominator is computed: *for every possible dataset*, it maximizes the likelihood. This means that a very flexible model, which can fit a wide variety of datasets, would overfit but also be strongly penalized.

Minimum description length then defines the *model complexity* as exactly the log of the denominator in Equation (3.12):

$$\mathrm{COMP}(\mathcal{M}; \pi) := \log \int \max_\theta p(y^n \mid \theta; \mathcal{M})\,, \pi(\theta) dy^n\,, \tag{3.13}$$

such that if we substitute p^{NML} into Equation (3.11), we can write the code length as

$$\begin{aligned}\mathrm{MDL}(y^n, \mathcal{M}_\pi) = &-\log \kappa(\mathcal{M}_\pi) + \sum_{\theta\in\Theta} -\log \pi(\theta) \\ &+ \sum_{i=1}^{n} -\log \max_\theta p(y^n \mid \theta; \mathcal{M})\pi(\theta) + n\,\mathrm{COMP}(\mathcal{M}; \pi)\,.\end{aligned} \tag{3.14}$$

The MDL principle states that we should select model with the lowest code length. The key differentiators between models are the last two terms of

8: In a Bayesian framework, *"with help from the prior"* would mean integrating over the prior to obtain the posterior distribution. In MDL it usually refers to the appearance of the prior in the normalized maximum likelihood (see Equation (3.12) below)

Domains must be countable

In Equation (3.10), we usually assume that the parameters θ and samples y_i take values on *countable* domains, to avoid dealing with probability densities.
Application of the MDL is already considerably more difficult when parameters take continuous values [48, p. 4.6.4].

9: Since devising minimal-length codes is a very hard problem in general, the first approach can leave some doubt that with a better code, we could obtain a lower MDL. Defining the code length as $-\log p$ avoids this problem. It's also a reasonable choice, since Shannon's source coding theorem guarantees that it is in some sense the asymptotically reachable lower bound. Note that when we do this, the problem of minimizing the code length becomes equivalent to minimizing the entropy. The only difference is that in MDL one typically uses a funny probability (the normalized maximum likelihood (Equation (3.12))) for p.

10: Grünwald and Roos [92] use the name "luckiness function", but this is really just another name for an improper prior.

If we use the model evidence as the universal distribution, MDL model selection is equivalent to comparing Bayes factors (p. 52) [92].

Note

Because the integrand in Equations (3.12) and (3.13) is refit to each y^n, it does not concentrate as strongly as the likelihood. Thus compared to other criteria, we need more data to invoke asymptotic approximations.

Equation (3.14): the fit to the data, and the model complexity. (The importance of the first two terms tends to be limited due either to the use of uniform priors, or the fact that they don't scale with n.)

In practice, the integral in Equation (3.13) is almost always hopelessly intractable. Computations in MDL therefore usually make strong asymptotic approximations; using MDL in practice requires knowing what approximations to use in which scenario, so that calculations terminate in finite time.

Insensitivity to the true distribution

Model complexity in MDL (Equation (3.13)) does not depend on the true data distribution: it seeks to minimize prediction error in a worst-case-over-all-data sense [92, p. 7]. Any mathematically possible dataset, however unlikely, is given equal weight.
Any suppression of implausible data points comes from the model not being able to fit them.
This makes MDL wholly unsuited for comparing learning machines designed to be universal function approximators.

MDL is inconsistent when models are misspecified

Similar to the case with Bayesian model evidence, when the true data-generating process is not included in the candidate models, the complexity term can overpenalize the optimal model (in the sense of smallest D_{KL} between it and the true model) and cause it to be rejected [93].

Sensitivity to approximations

Although the theoretical formalism of MDL is quite general, in practice one usually needs to invoke some approximation in order to carry out computations. These may impose additional restrictions—such as the likelihood being smooth, unimodal or positive definite—which are just as important in practice, if not in theory.

3.5. Information criteria

3.5.1. The basic program of information criteria

To develop an intuition for what information criteria try to achieve, let us consider a simple example where we have n samples $\{y_i \in \mathbb{R}^k\}_{i=1}^n$ from a k-dimensional uncorrelated Gaussian, so that the probability factorizes as:

$$p(y_i) = \prod_{l=1}^{k} \frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left(-\frac{(y_{i,l} - \mu_l)^2}{2\sigma_l^2}\right). \tag{3.15}$$

We assume to have two unfitted candidate models (i.e. two hypothesis classes) q and $\tilde{q}$. We want to know which of these is best supported by the observations $\{y_1, \dots, y_n\}$.

Let us further assume that the standard deviations σ_l are known[11] but the

Note

Some of the ideas in Section 3.5.1, mode and mean of the loss distribution, are based on an example in [76].

11: As we will see below, this example is not as artificial as it may initially seem, since around the maximum likelihood parameters, all *non-singular models* reduce to estimating a set of means with unit variance.

means μ_l must be estimated from data as $\hat{\mu}_{l|y^n} = \frac{1}{n}\sum_{i=1}^{n} y_{i,l}$. If we assume one of the models to be well-specified (as do many criteria; see Table 3.1 (p. 48)) then either q or $\tilde{q}$ is also an uncorrelated Gaussian. Without loss of generality we choose this to be q, with $\tilde{q}$ any other distribution with support on $\mathbb{R}^k$.

The loss, which we define as two times the negative log likelihood,[12] will depend on the data both through the y_i themselves, and through our estimate of the mean:

12: The factor two is included to match the *deviance*, defined below.

$$\begin{aligned} -2 \times \log \text{likelihood}(\hat{\mu}_{|y^n}) &= \sum_{i=1}^{n}\sum_{l=1}^{k} \log\left(2\pi\sigma_l^2\right) + \sum_{i=1}^{n}\sum_{l=1}^{k} \frac{(y_{i,l} - \hat{\mu}_{l|y^n}))^2}{\sigma_l^2} \\ \ell(\hat{\mu}_{|y^n}; y^n) &= \sum_{l=1}^{k} \underbrace{\left\{ n\log\left(2\pi\sigma_l^2\right) + \sum_{i=1}^{n}\left(\frac{y_{i,l} - \hat{\mu}_{l|y^n}}{\sigma_l}\right)^2 \right\}}_{\text{component loss}} \\ &=: \sum_{l=1}^{k} \ell_l(\hat{\mu}_{l|y^n}; y_l^n)\,. \end{aligned} \tag{3.16}$$

The independence of the components allows us to rewrite the total loss as a sum over component losses ℓ_l. Within each component loss, the first term in the sum is composed only of known parameters, and *if we estimate the mean correctly*—i.e. if $\hat{\mu}_{l|y^n} = \mu_0$—the second term is the sum of n standard normal variables. It would therefore follow a χ^2 distribution of degree n:

$$\begin{aligned} \ell_l(\hat{\mu}_{l|y^n}; y_l^n) &= n\log 2\pi\sigma_l^2 + \chi_n^2 \\ &= \text{cst} + \chi_n^2\,. \end{aligned} \tag{3.17}$$

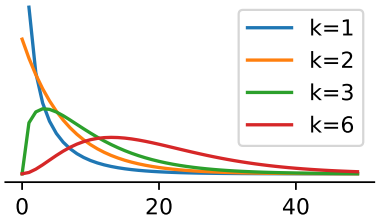


Figure 3.1.: χ_k^2-distributions with k degrees of freedom.

We can then obtain a distribution for the total loss by recalling that the sum of χ^2-distributions is also χ^2, with degrees of freedom equal to the sum of the degrees of freedom of the summands:

$$\ell(\hat{\mu}_{|y^n}; y^n) = \text{cst} + \chi_{nk}^2\,. \tag{3.18}$$

This describes the distribution of the loss, assuming the model assumptions are correct; we display the density function of χ_k^2 for a few different values of k in Figure 3.1.

Of course, we don't actually know the true mean μ, which is why we ask which of the two candidate models q and $\tilde{q}$ has the highest loss *asymptotically*, i.e. in the limit $n \to \infty$; as long as we have an *consistent estimator* (see above) for the mean, it will converge to the true value:

$$\lim_{n\to\infty} \hat{\mu}_{|y^n} = \mu\,. \tag{3.19}$$

Since the samples are i.i.d., this is also n times the expected loss of a *single* sample y:

$$\ell(\hat{\mu}_{|y^n}; y^n) \underset{n\to\infty}{\sim} \ell(\mu; y^n) \underset{n\to\infty}{\sim} n\mathbb{E}_{y\sim p}\big[\ell(\mu; y)\big] \tag{3}$$

(the symbol $\sim$ meaning that these three quantities become asymptotically equal as we increase n). This is useful because we can treat sums over samples as estimators for the expectation:

$$\sum_{i=1}^{n} \ell(\hat{\mu}_{|y^n}; y_i) = n\,\widehat{\mathbb{E}_{y\sim p}\big[\ell(\mu; y)\big]}\,. \tag{3.20}$$

However, even if the estimator $\hat{\mu}_{|y^n}$ of the mean is unbiased, the estimator for the *loss* will be biased if we use the same data y^n to compute both $\hat{\mu}_{|y^n}$ and $\ell(\hat{\mu}_{|y^n}; y_i)$: reusing data causes one to *overfit* and underestimate the loss of each model.

A crude way of estimating the amount of overfitting is to compare the mode and mean of the loss distribution: adjusting μ to maximize the likelihood on y^n will, by definition, yield a value corresponding to the *mode* of the loss distribution, but it is the *mean* that is an unbiased estimator of the expected loss. In this simple example, each component loss ℓ_l follows a χ^2_n, which has mode $n-2$ and mean n. Each component loss ℓ_l should therefore be, on average, 2 larger than its true mean:

$$\begin{aligned} \mathbb{E}_{y_l^n \sim p}\left[\ell_l(\hat{\mu}_{l|y^n}; y_l^n)\right] &= \mathbb{E}_{y_l^n \sim p}\left[\ell_l(\mu_l; y_l^n)\right] + 2\,, \quad \forall\, l = 1, \dots, k \\ &= n\, \mathbb{E}_{y_l \sim p}\left[\ell_l(\mu_l; y_l)\right] + 2\,, \end{aligned} \tag{3.21}$$

such that the total loss will be overestimated by $2k$:

$$\begin{aligned} \mathbb{E}_{y^n \sim p}\left[\ell(\hat{\mu}_{|y^n}; y^n)\right] &= \sum_{l=1}^{k} \mathbb{E}_{y_l^n \sim p}\left[\ell_l(\hat{\mu}_{l|y^n}; y_l^n)\right] \\ &= \sum_{l=1}^{k} \left\{ n\, \mathbb{E}_{y_l \sim p}\left[\ell_l(\mu_l; y_l)\right] + 2 \right\} \\ &= n\, \mathbb{E}_{y \sim p}\left[\ell(\mu; y)\right] + 2k\,. \end{aligned} \tag{3.22}$$

Exchanging terms then provides an unbiased estimate of the expected sample loss:

$$n\, \mathbb{E}_{y \sim p}\left[\ell(\mu; y)\right] = \mathbb{E}_{y^n \sim p}\left[\ell(\hat{\mu}_{|y^n}; y^n)\right] - 2k\,. \tag{3.23}$$

Note that k happens to be the number of location parameters the model estimates; as we will see, many comparison criteria indeed take the form

$$n\, \widehat{\mathbb{E}_{y \sim p}\left[\ell(\mu; y)\right]} = \ell(\hat{\mu}_{|y^n}; y^n) - 2 \cdot \#\text{effective parameters}\,, \tag{3.24}$$

with the definition of "effective parameters" depending on the criterion.

Terminology

Criteria in the form of Equation (3.24) are usually called *information criteria*, even if, like the BIC or WAIC, they account for a different type of overfitting

Important caveat

Correcting for overfitting in this way only holds *in expectation*: if we repeat the analysis on many different datasets, then *across those analyses* the loss we assign to each model will be unbiased, so that we can expect to select the true model more often than not. For any *specific* dataset however, the bias due to overfitting may be more or less than $2k$, so this does not guarantee that any particular comparison will select the correct model.

We emphasize here that **the whole program of information criteria is designed to determine the maximum likelihood model**. It is therefore imperative that the true distribution is actually within the hypothesis class.[13] Otherwise the optimal model (from the point of view of prediction) may be very different from the one that minimizes $\mathbb{E}_p[\text{loss}]$ [93]. It can also happen that the maximum likelihood model is suboptimal for other reasons – for example, another model may have slightly higher loss, but be more tolerant to parametrization errors. This is one reason it can sometimes be preferable

13: Technically, the justification for most criteria only requires the existence of a *pseudo*true model in the hypothesis class: a unique model that is closest to the true model in Kullback-Leibler divergence. This however only works well when the likelihoods are convex in parameter space [93, figure 4], which is not the case for most interpretable models.

to rank models using a different measure of predictive performance than the log likelihood.

3.5.2. Formalization of the correspondence to a χ^2 – Wilks' theorem and the likelihood-ratio test

Although in Section 3.5.1 (p. 56) we considered a specific example of a model where only the *location* parameters are unknown,[14] the appearance of a χ^2 distribution is in fact a general feature of comparisons between *nested*, *non-singular* models. This is a standard result known as *Wilks' theorem*, and it is the main theoretical justification for information criteria like the AIC. Thus, with some small exceptions, these criteria are only valid when Wilks' theorem is also valid.

14: A *location* parameter is a general name for a parameter that translates a probability distribution without changing its shape.

To formulate the result, call one model the *null model* (q_0, parameter space Θ_0), and the other the alternative model (q_A, parameter space Θ_A). We assume q_0 to be *nested* in q_A.

Terminology: Nested model

A model q_0 is said to be *nested* in another model q_A if it can be recovered from q_A by fixing some of its variables.
For example, $q_0(x) = Ax$ is nested in $q_A(x) = Ax + b$, since it can be recovered from the latter by setting $b = 0$.
We will use nested models in Section 5.2.9 (p. 119) when we compare different model selection methods to our own EMD.

Wilks' theorem

Let q_0 and q_A be two non-singular models, with q_0 nested in q_A. Let m be the number of parameters of q_0, and $s > m$ the number of parameters of q_A. Further assume that the MLE of each model does not lie on the boundary of their respective parameter spaces **and that q_0 is true**. Then,

$$2 \log \frac{q_A(y^n|\hat{\theta}_A)}{q_0(y^n|\hat{\theta}_0)} \sim \chi^2_{s-m}\,, \tag{3.25}$$

i.e. the l.h.s. follows a χ^2-distribution with number of degrees of freedom is equal to the difference in the numbers of parameters.
See [52, pp. 345 and 377] for the theorem statement and a proof sketch.

In other words, Wilks' theorem says that asymptotically, *any* parameter behaves like the location parameters $\hat{\mu}_l$ in Equation (3.17) (p. 57), as long as the requirements of the theorem are satisfied.

Caution

Although frequently taken as a matter of fact in the statistical literature, the requirements of Wilks' theorem are often not satisfied in practice.

As in the example above, this correspondence to a χ^2 provides a reference distribution to interpret values of the deviance; this is known in statistics as a *likelihood ratio test*. Since the χ^2 is derived assuming that the smaller model q_0 is true, the test is applied by treating q_0 as a the null hypothesis. If the value of the likelihood ratio has low probability under χ^2, that suggests that the additional parameters in q_A are not useless, so we reject q_0 and take q_A instead.

Information criteria take the simple form

$$B(\mathcal{M}) = \underbrace{-2 \times \log \text{likelihood}(\mathcal{M})}_{\text{loss } \ell} + \text{correction}(\mathcal{M}), \tag{3.26}$$

where the likelihood is evaluated on the fitted model, e.g. using the parameters estimated by maximum likelihood (MLE) or maximum a posteriori (MAP).

Note that *if q_0 is the true data-generating process*, the log likelihood ratio is closely related to the Kullback-Leibler divergence (D_{KL}) between the distributions:

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{q_0(y^n|\hat{\theta}_0)}{q_A(y^n|\hat{\theta}_A)} = \mathbb{E}_{y\sim q_0}\left[\log \frac{q_0(y|\hat{\theta}_0)}{q_A(y|\hat{\theta}_A)}\right] = D_{\mathrm{KL}}(q_0 \| q_A). \tag{3.27}$$

Recall that the Shannon entropy of a distribution is defined as [59, §2.4]

$$\mathbb{E}_p[\log p]; \tag{3.28}$$

when q_0 is the true model (so when $q_0 = p$), the D_{KL} can therefore be interpreted as the expected cost, in terms of entropy bits, of using q_A to approximate q_0.

When q_0 is *not* the true model, the D_{KL} can still be thought as a difference in bits, but it does not correspond to an average cost on the true data distribution.

This connection to information theory is why these are called "information criteria". Below we describe some of the most common criterion of this type, the *Akaike information criterion*.

3.5.3. Akaike information criterion (AIC)

For a model with k free parameters, the AIC takes the form

$$\mathrm{AIC}(q; y^n) = -2 \log q(y^n \mid \theta) + 2k. \tag{9}$$

This is exactly the estimator following from Equation (3.23) (p. 58); indeed, Akaike presents his derivation as an extension of the χ^2 correspondence to cases where the model q_0 is "not too false" [94, p. 604]. The starting point for his derivation is an expectation of the loss (here the log likelihood) over datasets ($\mathbb{E}_{y^n\sim p}$)—Akaike seems to be the one to have introduced this key idea for estimating generalization error [94, p. 601].

Like Wilks' theorem, the derivation of the AIC is heavily reliant on a first order Taylor approximation around the MLE. So although the models don't need to be nested [95], its applicability is also limited to non-singular models.

In brief

With nested models, the AIC selects the more complex one if there is enough data to fit its parameters without overfitting—irrespective of which model is "true".

Remark

There is a tension between this observation that the correction term is mostly relevant for small n, and the fact that the AIC only holds asymptotically for large n.

3.6. Conclusion

Part of the challenge in discussing model comparison methods is that it means different things for different authors. This makes the idea space difficult to

navigate. Key assumptions (like the presence of the true model within the hypothesis class) are often unstated and taken for granted, and when they are stated, the same words may have very different meanings to different authors—for example, "overfitting" can refer to a mistake either in the generalization or the information criteria paradigm. Of the three paradigms described above, and for the context we are interested in, the generalization error seems to unambiguously be the correct approach: it is the one consistent with the idea of scientific induction, as we explain in Section 2.1.4 (p. 34) and develop further in Chapter 5 (p. 99).

However, in order to properly navigate the idea space, and build appropriate connections to existing ideas and results, it is important to also understand the other main approaches based on model complexity and distance from true parameters. This is why, although the rest of our thesis will consider generalization error exclusively, in this chapter we devoted substantial space to describing those other methods, and establishing how they differ.

Learning models

Inference of a Mesoscopic Population Model from Population Spike Trains

4.

To understand how rich dynamics emerge in neural populations, we require models exhibiting a wide range of behaviours while remaining interpretable in terms of connectivity and single-neuron dynamics. However, it has been challenging to fit such mechanistic spiking networks at the single neuron scale to empirical population data. To close this gap, we propose to fit such data at a meso scale, using a mechanistic but low-dimensional and hence statistically tractable model. The mesoscopic representation is obtained by approximating a population of neurons as multiple homogeneous 'pools' of neurons, and modelling the dynamics of the aggregate population activity within each pool. We derive the likelihood of both single-neuron and connectivity parameters given this activity, which can then be used to either optimize parameters by gradient ascent on the log-likelihood, or to perform Bayesian inference using Markov Chain Monte Carlo (MCMC) sampling. We illustrate this approach using a model of generalized integrate-and-fire neurons for which mesoscopic dynamics have been previously derived, and show that both single-neuron and connectivity parameters can be recovered from simulated data. In particular, our inference method extracts posterior correlations between model parameters, which define parameter subsets able to reproduce the data. We compute the Bayesian posterior for combinations of parameters using MCMC sampling and investigate how the approximations inherent to a mesoscopic population model impact the accuracy of the inferred single-neuron parameters.

Published as a "Major advance article": Alexandre René, André Longtin, and Jakob H. Macke. "Inference of a Mesoscopic Population Model from Population Spike Trains". In: *Neural Computation* 32.8 (June 10, 2020)

4.1. Introduction

Neuron populations produce a wide array of complex collective dynamics. Explaining how these emerge requires a mathematical model that not only embodies the network interactions, but that is also parameterized in terms of interpretable neuron properties. Just as crucially, in order to draw data-supported conclusions, we also need to be able to infer those parameters from empirical observations. These requirements tend to involve a trade-off between model expressiveness and tractability. Low-dimensional state-space models [97–100] are simple enough to allow for inference, but achieve that simplicity by focussing on phenomenology: any mechanistic link to the individual neurons is ignored. Conversely, microscopic mechanistic models with thousands of simulated neurons do provide that link between parameters and output [42, 101]; however, this complexity makes the analysis difficult and limited to networks with highly simplified architectures [102, 103]. Since methods to fit these models to experimental data are limited to single neurons [38], it is also unclear how to set their parameters such that they capture the dynamics of large heterogeneous neural populations.

To reduce the problem to a manageable size and scale, one can consider models that provide a mesoscopic dynamical description founded on microscopic single-neuron dynamics [44, 104, 105]. Specifically, we will focus on the model described by Schwalger, Deger, and Gerstner [20], where neurons are grouped into putative excitatory (E) and inhibitory (I) populations in a cortical column. The key approximation is to replace each population with another of equal size, but composed of identical neurons, resulting in an effective mesoscopic model of homogeneous populations. In contrast with previous work on population rate dynamics [40, 105, 106], Schwalger, Deger, and Gerstner [20] correct their mean-field approximations for the finite size of populations. They are thus able to provide stochastic equations for the firing rate of each population with explicit dependence on the population sizes, neuron parameters, and connectivities between populations (Figure 4.1a, top). We use these equations to fit the model to traces of population activity.

Directly inferring mesoscopic model parameters has a number of advantages compared to extrapolating from those obtained by fitting a microscopic model. For one, it allows the use of data that do not have single-neuron resolution. In addition, since neuron parameters in a mesoscopic model represent a whole population, there may not be a clear way to relate micro- and mesoscopic parameters if the former are heteregeneous. By inferring population parameters from population recordings, we target the values that best compensate for the mismatch between the data and the idealized mesoscopic model (Figure 4.1b).

The method we present assumes that the model to be inferred can be expressed as a set of stochastic equations and that we have access to time series for both the observed (and possibly aggregated) neural activities and external input. It is thus not limited to mesoscale models, and could also be applied to e.g. Hodgkin-Huxley type neurons in isolation or networks. Nevertheless, in this paper, the underlying microscopic model does make the inferred parameters more readily interpretable, and provides a good idea of what values an inference algorithm should find for the parameters.

Approximate Bayesian computing (ABC) methods have recently been developed for inferring models where stochastic equations are treated as a black box simulator [107–110]. In such a case, one does not have access to the internal variables of the model and thus cannot compute the likelihood of its parameters; instead, these methods make use of repeated simulations to find suitable parameters. While this makes them applicable to a wide range of models, the repeated simulations can make them computationally expensive, and best suited to optimizing a set of statistical features rather than full time traces. Moreover, for the models of interest here, the likelihood can be derived from the stochastic evolution equations.

We show in this work that the likelihood can indeed be used to infer model parameters using non-convex optimization. The resulting optimization problem shares many similarities with training recurrent neural networks (RNNs) popular in machine learning [1, 111], and allows us to leverage optimization tools from that field. However, RNNs in machine learning are typically based on generic, non-mechanistic models, which implies that interpretation of the resulting network can be challenging (but see e.g. work on RNN visualization by Barak et al. [112–114]). Thus, our approach can be regarded as complementary to RNN approaches, as we directly fit a mechanistically interpretable model.

This paper is organized as follows. In Sections 4.2.1 and 4.2.2 we establish that maximum likelihood inference for our chosen mesoscopic model is sound, and in Section 4.2.3 provide empirical estimates for the amount of data this procedure requires. Using the example of heterogeneous populations, Section 4.2.4 then shows how inference can find effective parameters that compensate for the mismatch between data and model. In Section 4.2.5 we identify co-dependence between multiple model parameters by recovering the full Bayesian posterior. Finally, Section 4.2.6 demonstrates that the approach scales well by considering a more challenging four population model with thirty-six free parameters. Section 4.3 discusses our results, with an emphasis on circumscribing the class of models amenable to our approach. Method details are provided in Section 4.4, along with technical insights gained as we adapted likelihood inference to a detailed dynamical model. Additional details, including a full specification of parameter values used throughout the paper, are given in Appendices 4.A to 4.I.

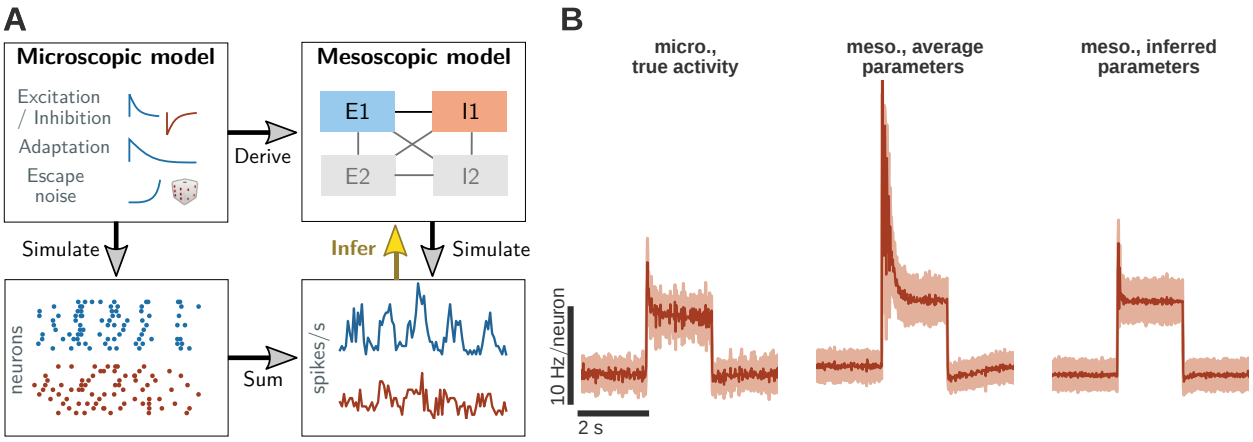


Figure 4.1.: a. General procedure to infer parameters of a mesoscopic population model from microscopic data. A microscopic model of GIF neurons is used to generate spike trains, which are averaged to obtain traces of population activity; these traces constitute our data. A mesoscopic model of either two or four populations is then fit to these traces. Simulating the mesoscopic model with the inferred parameters allows us to evaluate how well it reproduces the true dynamics. **b. For heterogeneous systems, average parameters might not predict mean activity.** Mean activity (line) and its standard deviation (shaded area) for a heterogeneous microscopic model (left) and mesoscopic models attempting to approximate it (middle, right). A mesoscopic model constructed by averaging parameters across the microscopic population overestimates the population's variability (middle). Inferred parameters in this case deviate from these averages and provide a better representation of the true activity (right). Models are as in Figure 4.5; traces are for the inhibitory population. Means and standard deviations are computed from 50 realizations and averaged over disjoint bins of 10 ms.

4.2. Results

4.2.1. Model summary

We studied the pair of microscopic and mesoscopic models developed by Schwalger, Deger, and Gerstner [20] and described in Sections 1.4.4 and 1.4.5, which are designed to represent excitatory (E) and inhibitory (I) populations of a putative cortical column of four neural layers [42]. For this study we only considered layers 2/3 and 4, and made minor parameter adjustments to maintain realistic firing rates (see Appendix 4.A). We also reduced all population sizes by a factor of 50 to ease the simulation of the microscopic model. This increases the variance of population activities, and so does not artificially simplify the task of inferring mesoscopic parameters.

The microscopic model is composed of either two or four populations of generalized integrate-and-fire (GIF) neurons. Neurons are randomly connected, with connectivity probabilities depending on the populations. The combination of excitatory and inhibitory input, along with internal adaptation dynamics, produces for each neuron i a time-dependent firing rate $\Lambda_i(t|\mathcal{H}_t)$; this rate is conditioned on the spike history up to t, denoted $\mathcal{H}_t$ (for equations see Section 4.4.1). Whether or not that neuron spikes within a time window $[t, t+\Delta t)$ is then determined by sampling a

Table 4.1.: Key variable definitions.

Variable	Definition
N_α	No. of neurons in population α.
M	No. of populations, $\alpha = 1, \dots, M$.
L	No. of time steps used to compute the likelihood.
Δt	Time step.
$\mathcal{I}_\alpha$	Set of indices of neurons belonging to population α.
$s_i(t)$	1 if neuron i spiked within time window $[t, t+\Delta t)$, 0 otherwise.
$A_\alpha(t)$	Activity in population α averaged over time window $[t, t+\Delta t)$.
$a_\alpha(t)$	Expectation of $A(t)$ conditioned on $\{A(t')\}_{t'<t}$.

Bernoulli random variable [20]:

$$s_i(t|\mathcal{H}_t) \sim \text{Bernoulli}(\Lambda_i(t|\mathcal{H}_t)\Delta t)\,, \tag{4.1}$$

where Δt is chosen such that $\Lambda_i(t|\mathcal{H}_t)\Delta t \ll 1$ is always true; we later refer to this stochastic process as *escape noise*. If all parameters are shared across all neurons within each population, we call this a *homogeneous* microscopic model. Conversely, we call a model *heterogeneous* if at least one parameter is unique to each neuron. We denote $\mathcal{I}_\alpha$ the set of indices for neurons belonging to a population α.

The *expected activity* a_α of a population α is the normalized expected number of spikes,

$$a_\alpha(t|\mathcal{H}_t) = \frac{1}{N_\alpha}\sum_{i\in\mathcal{I}_\alpha}\Lambda_i(t|\mathcal{H}_t)\,, \tag{4.2}$$

which is a deterministic variable once we know the history up to t. In contrast, the *activity* A_α of that population is a random variable corresponding to the number of spikes actually observed,

$$A_\alpha(t|\mathcal{H}_t) \coloneqq \frac{1}{N_\alpha}\sum_{i\in\mathcal{I}_\alpha} s_i(t|\mathcal{H}_t)\,. \tag{4.3}$$

In practice data is discretized into discrete time steps $\{t_k\}_{k=1}^L$, which we assume to have uniform lengths Δt and to be short enough for spike events of different neurons to be independent within one time step (this condition is always fulfilled when the time step is less than the synaptic transmission delay). Under these assumptions, Equation (4.3) can be approximated by a binomial distribution [20],

$$A_\alpha^{(k)} \coloneqq A_\alpha(t_k|\mathcal{H}_{t_k}) \sim \frac{1}{N_\alpha\Delta t}\,\text{Binom}(N_\alpha a_\alpha(t_k|\mathcal{H}_{t_k})\Delta t;\; N_\alpha)\,. \tag{4.4}$$

If we repeat a simulation R times with the same input, we obtain an ensemble of histories $\{\mathcal{H}_{t_k}^r\}_{r=1}^R$ (due to the escape noise). Averaging over these histories yields the *trial-averaged activity*,

$$\bar{A}_\alpha^{(k)} \coloneqq \frac{1}{R}\sum_{r=1}^{R} A_\alpha(t_k|\mathcal{H}_{t_k}^r)\,, \tag{4.5}$$

the theoretical counterpart to the peristimulus time histogram (PSTH).

For the **microscopic model**, the history is the set of all spikes,

$$\mathcal{H}_{t_k} = \{s_i(t_l)\}_{\substack{i=1\ldots N\\ t_l<t_k}}\,. \tag{4.6}$$

To generate activities, we first generate spikes with Equation (4.1) and use Equation (4.3) to obtain activities (see Figure 4.1a).

For the **mesoscopic model**, hereafter referred to as "mesoGIF", the history only contains population activities:

$$\mathcal{H}_{t_k} = \{A_\alpha^{(l)}\}_{\substack{\alpha=1\ldots M\\ t_l<t_k}}\,. \tag{4.7}$$

The expected activity is then an expectation over all spike sequences consistent with that history, for which a closed form expression was derived in Schwalger, Deger, and Gerstner [20] (the relevant equations are given in Appendix 4.E). Activities are generated by using this expression to compute $a_\alpha(t)$ and then sampling Equation (4.4). Unless mentioned otherwise, for the results reported in the sections below we used the microscopic model for data generation and the mesoscopic model for inference.

In addition to homogeneity of populations and independence of spikes within a time step, the mesoscopic model depends on one more key approximation: that neuron populations can be treated as *quasi-renewal* (see Section 1.4.5,

as well as references [20, 41]). If neurons are viewed as having both refractory and adaptation dynamics, this is roughly equivalent to requiring that the latter be either slow or weak with respect to the former. (A typical example where this approximation does not hold is bursting neurons [41].) As described in Section 1.4.5, under these approximations the unbounded history $\mathcal{H}_{t_k}$ can be replaced by a finite state vector $S^{(k)}$ of *free* and *refractory* neurons, which is updated along with the expected activity $a^{(t)}$ (see Section 4.4.2). Since the update equations only depend on $S^{(k-1)}$, they are then Markovian in S. This in turn allows the probability of observations $P\left(A^{(L)}, A^{(L-1)}, \dots, A^{(1)}\right)$ to be factorized as $P\left(A^{(L)}|S^{(L)}\right) \cdot P\left(A^{(L-1)}|S^{(L-1)}\right) \cdots P\left(A^{(1)}|S^{(1)}\right)$, which is key to making the inference problem tractable.

4.2.2. Recovering population model parameters

We first consider a two-population model composed of E and I neurons. We use the homogeneous microscopic model to generate activity traces (Figure 4.2a), with a frozen noise input that is shared within populations; this input is sine-modulated to provide longer term fluctuations (c.f. Equation (4.36)). A maximum a posteriori (MAP) estimate $\hat{\eta}_{\mathrm{MAP}}$ of 14 model parameters is then obtained by performing stochastic gradient descent on the posterior (see Section 4.4). Because the likelihood is non-convex, we perform multiple fits, initializing each one by sampling from the prior (Figure 4.2b). We then keep the one that achieves the highest likelihood, which in practice is often sufficient to find a near-global optimum [115].

It is important to remember that one can only fit parameters that are identifiable given our data (recall Section 1.3). For example, in the mesoGIF model, the firing probability is determined by the ratio (see Equation (4.16))

$$\frac{u(t) - \vartheta(t)}{\Delta_{\mathrm{u}}}, \tag{4.8}$$

where u is the membrane potential, ϑ the firing threshold and Δ_{u} a parameter describing the level of noise. All of these quantities are computed in units of millivolts, and the terms in the numerator depend on the resting potential u_{rest} and threshold u_{th}. However, since Equation (4.8) is dimensionless, the choice of millivolts is arbitrary: after changing Δ_{u}, one can rescale u_{rest} and u_{th} (along with the synaptic weights w and reset potential u_{r}) to recover exactly the same dynamics. The set of parameters w, Δ_{u}, u_{rest}, u_{th} and u_{r} is thus degenerate, and they cannot all be inferred simultaneously; for this paper, we set the voltage scale to millivolts by fixing u_{rest} and u_{th} to the values proposed by Schwalger, Deger, and Gerstner [20]. Other parameters are similarly ill-constrained, and in total we inferred 14 model parameters; these are listed in Table 4.2.

We tested the inferred model on frozen low-pass-filtered white-noise of the same form as in Augustin et al. [116] (Figure 4.2c, top), ensuring that a range of relevant time scales are tested. Despite the frozen input, variability between realizations does remain: for the GIF model this is due to sampling the escape noise (Equation (4.1)), while for the mesoGIF model it is due to sampling the binomial in Equation (4.4). We thus compare models based on the statistics of their response rather than single realizations: each model is simulated 100 times with different internal noise sequences (for each neuron in the case of the GIF model, and for each population in the case of the mesoGIF model) to produce an ensemble of realizations, from which we estimate the time-dependent mean and standard deviation of $A(t)$. Mean and standard deviation are then averaged over disjoint 10ms windows to reduce variability due to the finite number of realizations. The results are reported as respectively lines and shading in Figure 4.2c, and show agreement between true and inferred models; we also find good agreement in the power spectrum of the response to constant input (Figure 4.3). Parameterizations for the training and test inputs are given in Section 4.4.8, and the full set of fits is shown in Figure 4.12.

4.2.3. Quantifying data requirements

While simulated data can be relatively cheap and easy to obtain, this is rarely the case of experimental data. An important question therefore is the amount required to infer the parameters of a model. To this end, we quantify in Figure 4.4 the accuracy of the inferred dynamics as a function of the amount of data.

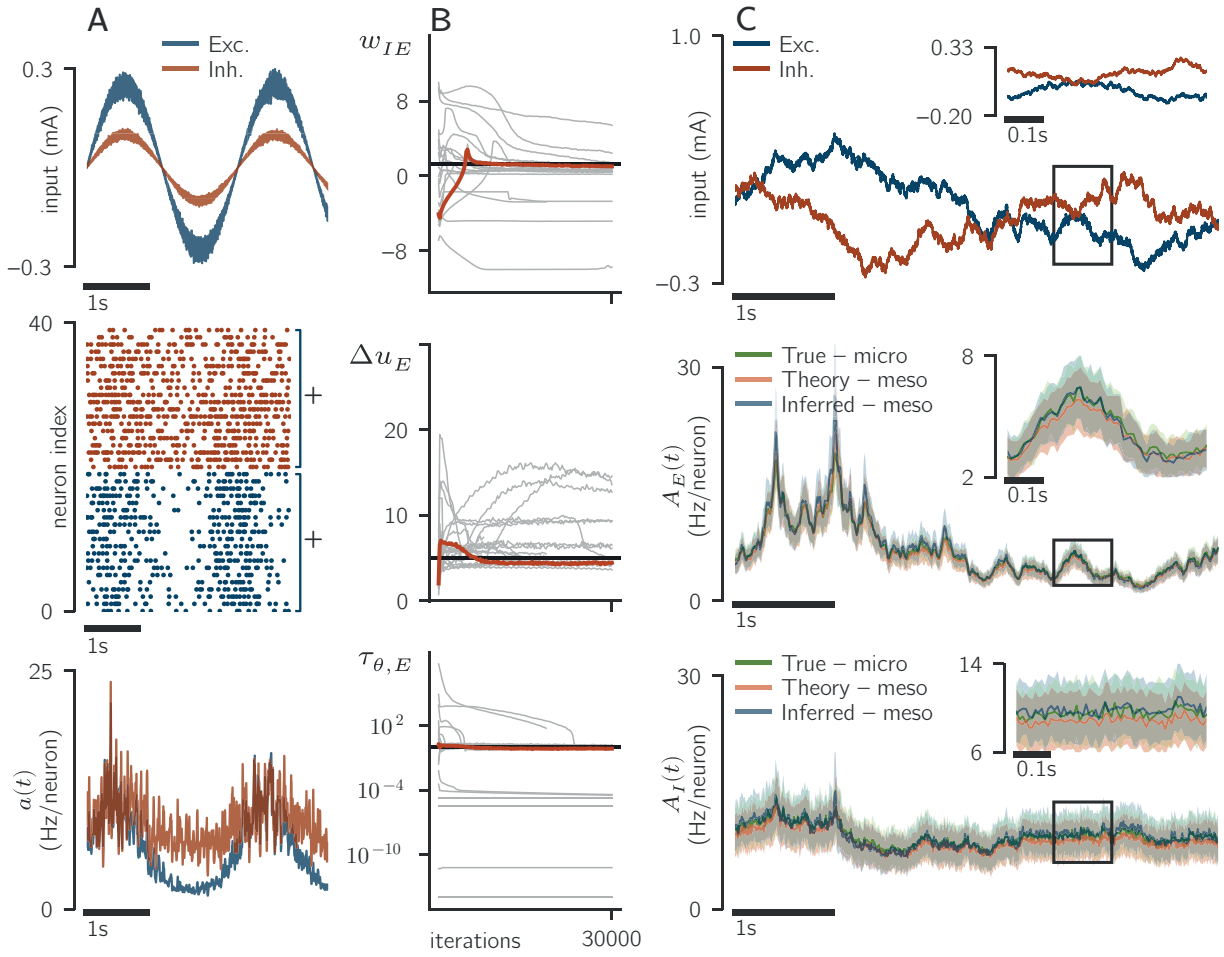


Figure 4.2.: Inferred model generalizes to different inputs. a. Data generation. Microscopic E and I populations receive a noisy sinusoidal input (Equation (4.36), Table 4.5), which is shared across populations (top). Generated spikes (middle) are summed across each population, such that the inference algorithm sees only the total activity in each. Despite being deterministic given the history $\mathcal{H}$, the population-averaged expected activity (Equation (4.2)) still shows substantial fluctuations due to stochasticity of the history itself (bottom). **b. Inference recovers parameter values close to those used to generate the data.** We performed a total of 25 fits, retaining the one that found the local optimum with the highest likelihood (shown in red). Black lines indicate the prediction of the mesoscopic theory of Schwalger, Deger, and Gerstner [20], based on ground truth values of the microscopic model. Fits for all 14 parameters are shown in Figure 4.12. **c. Inferred mesoscopic model reproduces input-driven variations in population activity.** For testing we used low-pass-filtered frozen white noise input (Equation 4.37, Table 4.6) (top) to simulate the inferred mesoscopic model; middle and bottom plots respectively show the activity of the E and I populations. Each model was simulated 100 times; we show the mean and standard deviation over these realizations as lines and shading of corresponding colors. (Values were averaged over disjoint bins of 10 ms.) Performance measures are $\bar{\rho} = 0.950, 0.946, 0.918$ and RMSE $= 3.42 \pm 0.07, 3.55 \pm 0.09, 3.40 \pm 0.08$ for the true, theory and inferred models respectively (see Section 4.4.7).

In order to be certain our ground truth parameters were exact, for this section we used the mesoGIF for both data generation and inference. This allows us to quantify the error on the inferred parameters, rather than just on the inferred dynamics. In a more realistic setting, data and model are not perfectly matched, and this will likely affect data requirements. Testing and training were done with different external inputs to avoid overfitting; as in Section 4.2.2, we used a sinusoidal frozen white noise for training and a low-pass-filtered frozen white noise for testing. During training, E and I neurons had respective average firing rates of 5.9 and 8.4 Hz, which translates to approximately 3500 spikes per second for the whole population.

We measured the accuracy of inferred dynamics by simulating the model with both the ground truth and inferred parameters, generating 20 different realizations for each model. These were used to calculate both the per-trial and trial-averaged Pearson correlation (ρ, $\bar{\rho}$) and root-mean-square error (RMSE, $\overline{\text{RMSE}}$) between models. An additional

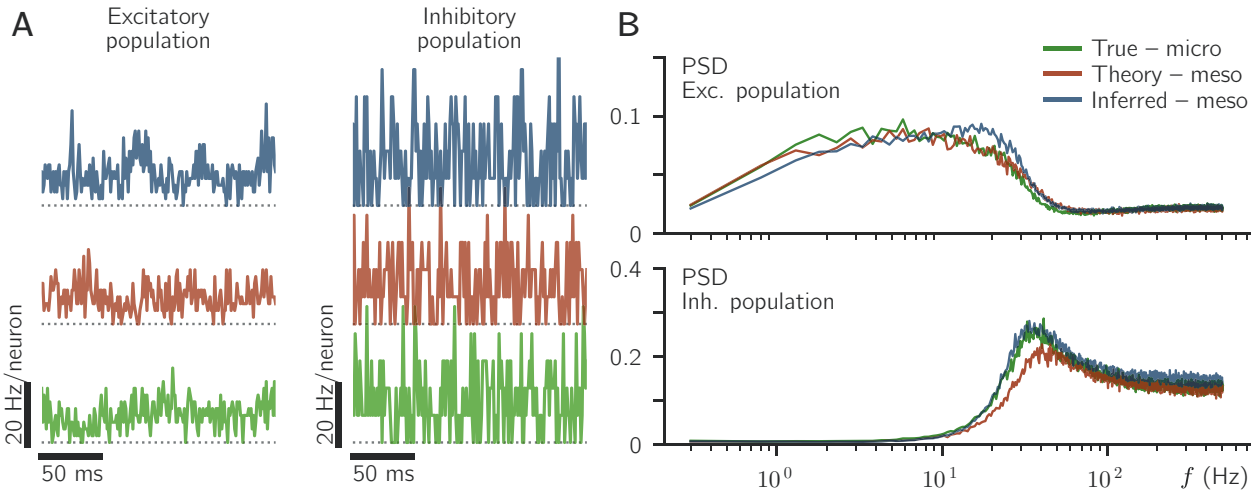


Figure 4.3.: Inferred model reproduces expected power spectral density. a. Segment of simulations of the same three models shown in Figure 4.2c under constant 0.5 mA input to both E and I populations. Dotted line indicates ordinate zero. **b.** Power spectral density for the excitatory (top) and inhibitory (bottom) populations. For each model, spectra were computed for 50 distinct realizations of 9s each and averaged. To reduce the error due to finite number of realizations, the frequency axis was then coarsened to steps of 0.5 Hz by averaging non-overlapping bins.

20 simulations of the ground truth model were used to estimate the best achievable performance for each measure. For per-trial measures, the reported standard deviation provides an estimate of the variability between realizations; for trial-averaged measures, the standard deviation is obtained by bootstrapping, and is purely an uncertainty on the statistic (it vanishes in the limit of large number of realizations). The calculations for these measures are fully described in Section 4.4.7. In subsequent sections, we report only the values of $\bar{\rho}$, RMSE to avoid redundant information.

Consistent with the observations of Augustin et al. [116], we found that ρ (in contrast to $\bar{\rho}$) does not allow to differentiate between models close to ground truth. The RMSE and $\overline{\text{RMSE}}$ on the other hand showed similar sensitivity, but may be unreliable far from ground-truth (as evidenced by the data point at L=1.25s in Figure 4.4c). Since the per-trial RMSE additionally quantifies the variability between realizations (through its standard deviation), we preferred it over its trial-averaged analog.

As we would expect, the inferred model better reproduces the dynamics of the true model when the amount of data is increased (Figure 4.4); when fitting all 14 parameters of the mesoGIF model, the inferred model no longer improves when more than 5–7 s of data are provided (Figures 4.4b–4.4c) – corresponding to a total of about 17 500–24 500 spikes. In Appendix 4.D, we repeat the test described here with smaller parameter sets (achieved by clamping certain parameters to their known ground truth values). We find that this has only a modest effect on the achieved performance, but does significantly improve the consistency of fits (compare Figures 4.9b and 4.9c). Inferring larger parameter sets is thus expected to require more fits (and consequent computation time) before a few of them find the MAP. Certain parameters are also more difficult to infer: for the case shown in Figure 4.4, relative errors on the inferred parameters range from 5% to 22% (see Appendix 4.D, Table 4.12). Parameters describing the inhibitory population ($\tau_{m,I}$, w_{IE}, w_{II}) show the highest relative error, as well as the escape rates (c_E, c_I) and the adaptation time constant ($\tau_{\theta,E}$).

4.2.4. Modelling high-dimensional heterogeneous populations with an effective low-dimensional homogeneous model

A frequently understated challenge of meso- and macroscale population models is that of choosing their parameters such that the dynamics of the modeled neuron populations are consistent with the high-dimensional dynamics of networks of individual neurons. A typical approach, when measurements of microscopic single neuron parameters are available, is to assign each parameter its mean across the population [8, §12]. However, as alluded to in Section 4.1,

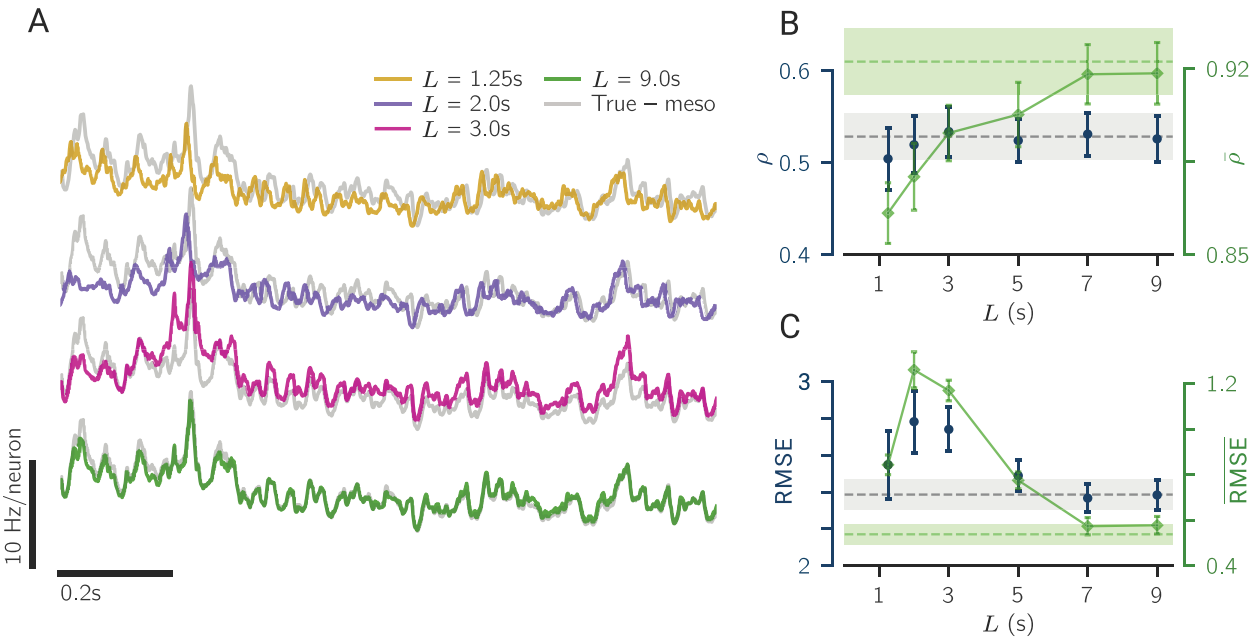


Figure 4.4.: Inferred model no longer improves after 20 000 spikes. Model parameters were inferred from data generated using shared frozen noisy sinusoidal input and tested on low-pass-filtered frozen white noise. **a. Sample portion of the simulated traces used to compute discrepancy measures.** Traces of the expected activity $a(t)$ of the excitatory population in a two population E-I model, using parameters inferred from increasing amounts L of data; all simulations are done on test input using the same random seed to sample the binomial in Equation (4.4). Note that the model did not see this input during training. **b,c. Inference performance of the inferred model.** Inferrence performance, measured as either Pearson correlation ρ (b) or RMSE (c) between 20 simulations of the inferred and true mesoscopic models. Dashed lines indicate maximum achievable performance, estimated by computing the measures on a different set of 20 realizations of the ground truth model; shading indicates standard deviation of that value. Blue points: per-trial statistics (Equations (4.32) and (4.33)); green points: trial-averaged traces (Equations (4.34) and (4.35)). Trial-averaged errors were estimated by bootstrapping. Results suggests that performance is well summarized by $\bar{\rho}$ and RMSE.

mean parameters do not always make good predictors for nonlinear systems; this is evidenced by Figure 4.5, which expands upon Figure 4.1b.

An alternative approach would be to fit the population model to observed population activities, such as to ensure maximum consistency with data—for example, by finding the maximum a posteriori (MAP) parameters. In this way we hope to find effective parameters that compensate for the mismatch between data and population model. *

To show that this can work, we made the microscopic model heterogeneous in three parameters: $\tau_{m,E}$, $\tau_{m,I}$ and $\tau_{\theta,E}$. These parameters were set individually for each neuron by sampling from a log-normal distribution (Figure 4.5a, Table 4.8). As in previous sections, output from the microscopic model under sine-modulated frozen white noise input was then used to train the mesoscopic one. For testing we used a single step input (Figure 4.5b); this allowed us to test the performance of the inferred model both in the transient and steady-state regimes. The per-trial RMSE and trial-averaged correlation $\bar{\rho}$ were computed on ensembles of realizations, as described in Section 4.4.7.

We considered three sets of parameters for the mesoscopic model. For the first, we set $\tau_{m,E}$, $\tau_{m,I}$ and $\tau_{\theta,E}$ to their sample averages. This produced rather poor results (Figure 4.5c, left); in particular, the transient response to the step is much more dramatic and long-lived than that of the ground truth model. As the neural model is highly nonlinear in its parameters, linearly averaging parameters is not guaranteed to produce optimal results.

The test results are improved when the heterogeneous parameters are inferred (Figure 4.5c, middle). However, fitting only the heterogeneous parameters gives the mesoscopic model only three degrees of freedom to compensate for approximating a heterogeneous model by a homogeneous one, and it still produces traces with too high variance.

* We should not expect to be able to fully compensate the mismatch with effective parameters. However, given the strong nonlinearities involved, it is reasonable to assume that we can do better than simply averaging the microscopic parameters, which would otherwise be the standard approach.

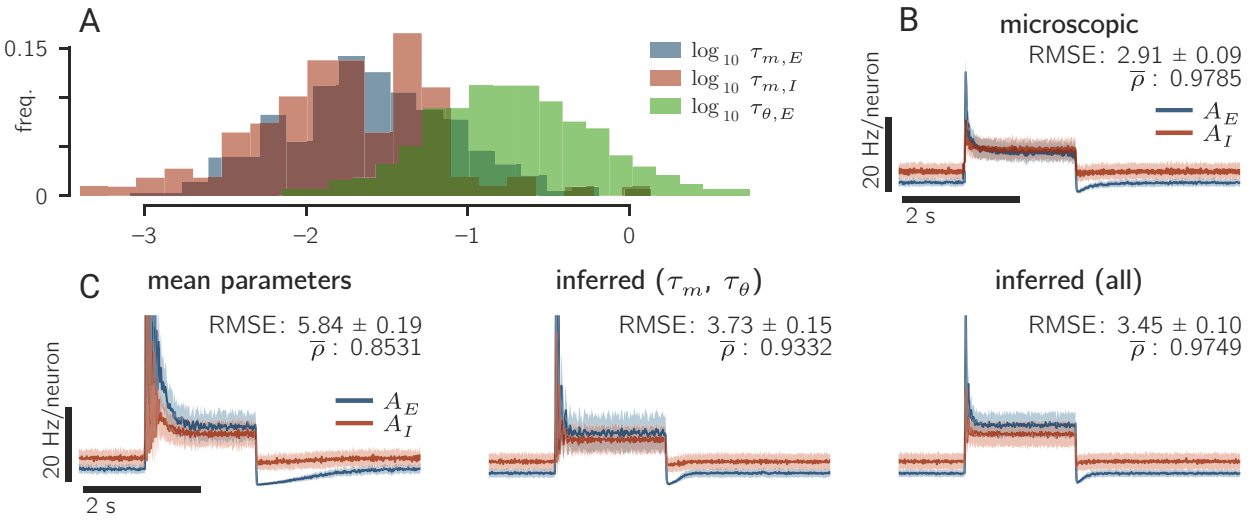


Figure 4.5.: Inferred effective parameters can compensate for mismatch between microscopic and mesoscopic models. a. A heterogeneous microscopic model of two populations was constructed by sampling three time constants from log-normal distributions (see Table 4.8). All other parameters are as in Section 4.2.2 and Figure 4.2, and the same sine-modulated white noise as in Figure 4.2a was used to train the model. **b. Heterogeneous microscopic model driven by a single step current.** Shown are the mean (line) and standard deviation (shading) of the model's response, computed from 60 realizations and averaged over disjoint windows of 10 ms. Realizations differ due to sampling the escape noise. **c.** Simulations of the mesoscopic model with the same step input as in (b), using mean parameters (left), inferred τ_m and τ_θ (middle – all other parameters homogeneous and set to ground truth), and the inferred full (14) parameter set (right). Line and shading have the same meaning as in (b) and are based on 50 realizations for each model; these differ by the sampling of the binomial in Equation (4.4). We see that inferred models more closely reproduce the trace in (b), which is confirmed by the decreased RMSE and increased $\bar{\rho}$.

Indeed, giving the model full freedom over the parameters provides another step improvement (Figure 4.5c, right), with output from the mesoscopic model differing from the target output only by a higher transient peak and slightly different mean activities (obtained parameter values are listed in Table 4.9). Thus while fitting more parameters may incur additional computational cost (Appendix 4.D), it also provides more opportunities to accommodate model mismatch.

The results of this section show the necessity of inferring population parameters rather than simply averaging single neuron values. It also demonstrates the ability of population models to reproduce realistic activities when we provide them with good effective parameters; in order to compensate for modelling assumptions, those parameters will in general differ from those of a more detailed microscopic model.

4.2.5. Full posterior estimation over parameters

It can often be desirable to know which parameters, or combinations of parameters, are constrained by the data. Bayesian inference, i.e. estimation of the posterior distribution over parameters given the data, can be used to not only identify the 'best-fitting' parameters, but also to characterize the uncertainty about these estimates. Notably, these uncertainties may be highly correlated across parameters: For instance, one expects an increase in E connectivity to compensate a decrease in (negative) I connectivity to the same population, and this is confirmed by the correlation in the marginals shown in Figure 4.6a. Interestingly, this correlation is in fact stronger for connectivities sharing the same *target* than those sharing the same *source*. More novel structure can be learned from Figure 4.6b, such as the strong correlation between the adaptation parameters, or the complete absence of correlation between them and the synaptic parameters. In particular, the tight relationship between $J_{\theta,E}$, $\tau_{\theta,E}$ and c_E suggests that for determining model dynamics, the ratios $J_{\theta,E}/\tau_{\theta,E}$ and $J_{\theta,E}/c_E$ may be more important than any of those three quantities individually.

Since there are 14 unknown parameters, the posterior is also 14-dimensional; we represent it by displaying the joint distributions between pairs, obtained by marginalizing out the other 12 parameters (see Section 4.4.6). Training data here were generated in the same way as in Section 4.2.2, from a homogeneous microscopic model with the

parameters listed in Table 4.2. To provide a sense of scale, we have drawn ellipses in Figure 4.6 to indicate the volume corresponding to two standard deviations from the mean under a Gaussian model. In a number of cases it highlights how the true distribution is non-Gaussian – for example the distributions of c_E, $J_{\theta,E}$ and $\tau_{\theta,E}$ are noticeably skewed.

A naive way to compute these 2D marginals would be to numerically integrate the likelihood; however, given that that leaves 12 dimensions to integrate, such an approach is computationally unfeasible. Instead we used Hamiltonian Monte Carlo (HMC) sampling [63, 117]. Monte Carlo methods are guaranteed to asymptotically converge to the true posterior—a valuable feature when one wishes to deduce interactions between parameters from its structure. Nevertheless, due to the complexity of mesoGIF's likelihood, memory and computational cost still required special consideration (see Section 4.4.6).

We note that the 2σ ellipses in Figure 4.6, while informative, are imperfect indicators of the probability mass distribution. If the posterior is Gaussian, then each projection to a 2D marginal places 86.5% of the probability mass within the ellipse; however for non-Gaussian posteriors this number can vary substantially. Moreover, the markers for ground truth parameters shown in Figure 4.6 may differ from the effective parameters found by the model (see Section 4.2.4).

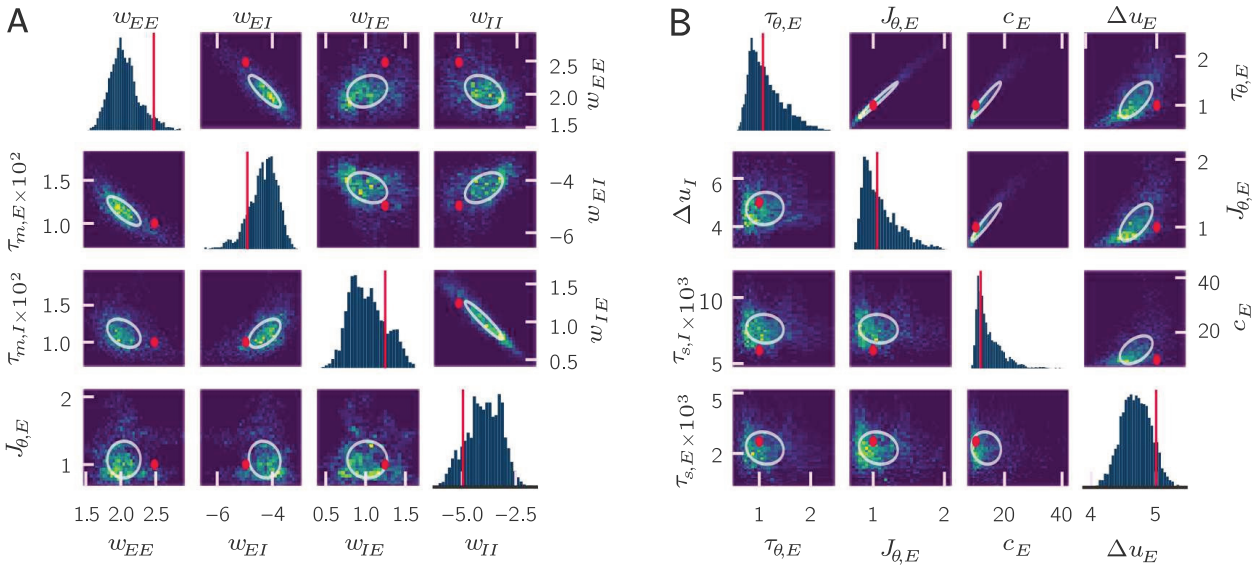


Figure 4.6.: Posterior probability highlights dependencies between model parameters. Panels show one and two-parameter marginals; all panels within a column use the same parameter for their abscissa. **a.** Above diagonal: Full posterior over the connectivities w. Strongest (anti)correlation is between pairs impinging on the same population (i.e. w_{EI}–w_{EE} and w_{IE}–w_{II}.) Below diagonal: Membrane time constants and adaptation strength show correlations with connectivity. Panels on the diagonal show the marginal for that column's parameters. Red dot or line shows the parameters' ground truth values. Ellipse is centered on the mean and corresponds to two standard deviations under a Gaussian model. The full posterior over all 14 parameters is shown in Figure 4.11 and was obtained with HMC sampling using data generated with the two-population homogeneous microscopic model. **b.** Above diagonal: Tight correlation between $\tau_{\theta,E}$, $J_{\theta,E}$ and c_E suggests their ratios are most important to determining model dynamics. Below diagonal: There is little correlation between adaptation and synaptic parameters. Diagonal panels, red marks and ellipses are as in a.

4.2.6. Pushing the limits of generalization

The previous sections have shown that we can recover 14 parameters of the two population model mesoGIF model. A natural question is whether this approach scales well to larger models. We investigated this by considering four neuron populations representing the L2/3 and L4 layers of the Potjans-Diesmann micro-circuit [42]. The associated higher-dimensional set of mesoscopic equations follow the same form as in previous sections [20]. There are 36 free parameters in this model, of which 16 are connectivities; they are listed in Table 4.2. Similar to previous sections, we trained mesoGIF on output from the microscopic model with sinusoidal drive (Figure 4.7a).

The L4 populations tend to drive the activity in this model, and we found that we do not need to provide any input to the L2/3 neurons to get parameter estimates that accurately predict population activity (Figure 4.8, left): the small fluctuations in L2/3 (Figure 4.7b) suffice to provide constraints on those population parameters. Those constraints of course are somewhat looser, and in particular connection strengths onto L4 are not as well estimated when compared to ground truth (Table 4.10).

Pushing the mesoscopic approximation beyond its validity limits using inputs with abrupt transitions understandably increases the discrepancy between ground truth and model (Figure 4.8, right). Indeed, such a strong input may cause neurons to fire in bursts, thereby breaking the quasi-renewal approximation (see Section 4.2.1). During an input spike, the true model shows small oscillations; the theoretical mesoGIF reproduces these oscillations but with an exaggerated amplitude and higher variance between realizations, and in contrast to Section 4.2.4, the inferred model does no better. This larger discrepancy with the true model is reflected in the performance measures (see Tables 4.14 and 4.15), and is consistent with the observation that the mesoGIF has higher variance during bursts [20, p. 15]. Slower time-scale dynamics are still accurately captured by both the theoretical and inferred models.

The capacity of the inferred model to generalize to unseen inputs is thus quite robust, with discrepancies between inferred and ground truth models only occurring when the test and training input were very different. Of course this is in part due to mesoGIF being a good representation of the activity of homogeneous GIF neurons: while inference may compensate for some discrepancies between the model and the data, it still can only work within the freedom afforded by the model.

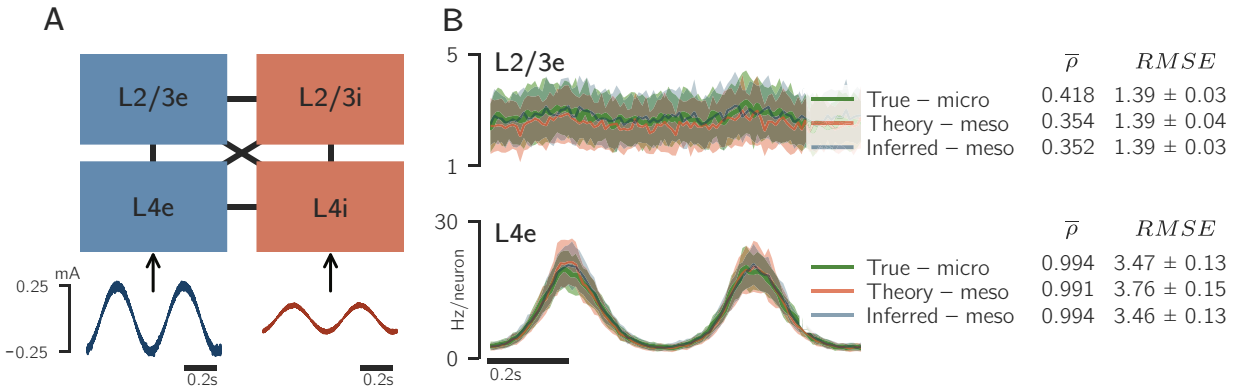


Figure 4.7.: Inference of a four population model with 36 free parameters. a. Model represented E (blue) and I (red) populations from layers L2/3 and L4 of a cortical column. During training, only L4 populations received external sinusoidal input. The homogeneous microscopic model was used to generate data. **b.** The mesoscopic model matches aggregate microscopic dynamics ("True – micro"), both when using theoretical ("Theory – meso") and inferred parameters ("Inferred – meso"). In contrast to the previous section, correlation and RMSE scores are reported separately for each population; they are computed from 60 realizations of each models.

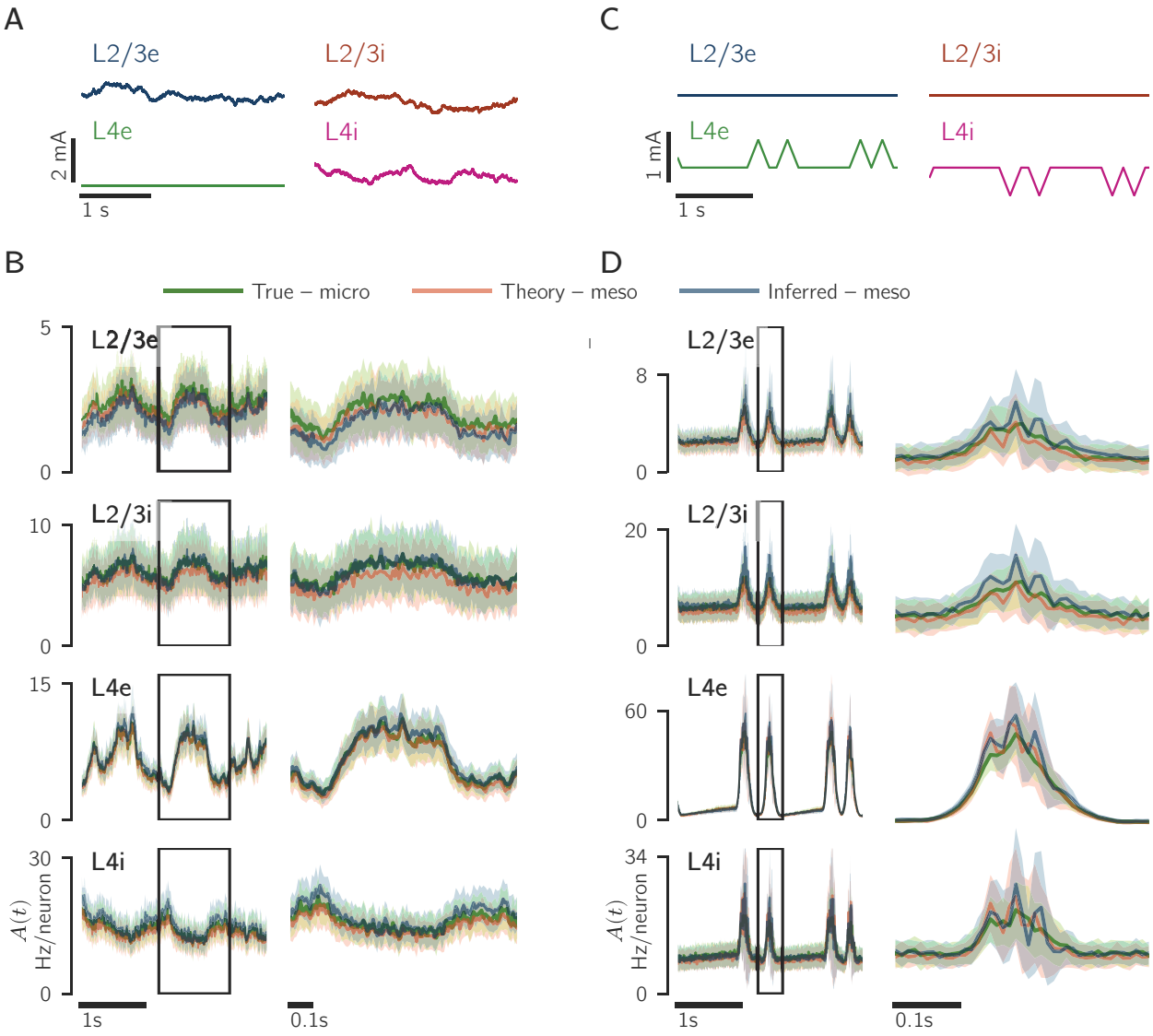


Figure 4.8.: Generalization errors appear with large deviations from the training input. We test the 36 parameter model inferred in Figure 4.7 under two different stimulation protocols. Lines and shading show mean and standard deviation over 60 realizations, computed as in Section 4.2.2. **a,b.** After completely removing external inputs to L4e (compare a with the training input in Figure 4.7a), predictions of the inferred and theoretical models are still indistinguishable. **c,d.** To obtain visible deviations between inferred and theoretical models, we used inputs (c) that violate the mesoGIF assumptions. Oscillations are present in both the microscopic and mesoscopic models, but in the latter have much larger amplitudes: compare the blue and red traces to the thicker green trace in d.

4.3. Discussion

Population models play a key role in neuroscience: they may describe experimental data at the scale they are recorded and serve to simplify the dynamics of large numbers of neurons into a human-understandable form. These dynamics may occur on a range of scales—from the mesoscopic, limited to a single cortical column, to the macroscopic, describing interactions between regions across the entire brain. Mechanistic models allow us to bridge those scales, relating micro-scale interactions to meso- or macro-scale dynamics; of these, the model chosen for this study allows for rich dynamics at the single level by including synaptic, refractory and adaptation dynamics.

We have demonstrated that it is possible to fit a mechanistic population model to simulated data by maximizing the likelihood of its parameters, in much the same way as is already done with phenomenological models [97–99].

Since mechanistic models describe concrete, albeit idealized, biophysical processes, they have the additional benefit that their parameters can be understood in terms of those processes. Moreover, those parameters are typically not dependent on the applied input, and thus we can expect the inferred model to generalize to novel stimulus conditions.

We also found that after making a few parameters heterogeneous, averaging did not recover the most representative parameters. In general, when there is discrepancy between model and data, the effective parameters are difficult to recover analytically – data-driven methods then provide a valuable supplement to theoretical analysis, in order to ensure that a model actually represents the intended biological process. Nevertheless, since the inference procedure is agnostic to the model, it is up to the modeler to choose one for which the effective parameters remain interpretable.

The approach we have presented requires only that a differentiable likelihood function be available, and thus is not limited to neuron population models. Stochastic models of neuron membrane potentials [118], of animal populations [119] and of transition phenomena in physics and chemistry [120, §7] are examples for which parameters could be inferred using this approach.

In practice we expect some models to be more challenging than others. For instance, evaluating the likelihood of a spiking model typically involves integrating over all time courses of the subthreshold membrane potential compatible with the observed spike train [121]. This integral can be difficult to evaluate accurately, especially for models incorporating adaptation and refractoriness [122, 123]. If evaluation of the likelihood is prohibitively expensive, likelihood-free approaches might be more appropriate [109, 110].

Of note also is that we required the dynamics to be formulated as a Markov process to express the likelihood (see Section 4.4.3). We achieved this by constructing a state vector, but the size of this vector adds substantial computational cost and in practice there is a trade-off between the length of the integration time window and the number of units (here neuron populations) we can infer. Since neural field models are also computationally represented by long state vectors, inference on these models would be subject to a similar trade-off. Finally, our current implementation assumes that the state S (see Section 4.4.2) can be fully reconstructed from observations. If only a partial reconstruction of S is possible, undetermined components of S form a latent state that must be inferred along with the parameters. This type of problem has already been studied in the context of dimensionality reduction [98, 124, 125], and it is conceivable that such methods could be adapted to our framework. Such an approach would allow one to perform dimensionality reduction with mechanistic models of temporal dynamics.

The work of Rule et al. [125] presents an interesting complement to ours. The authors therein consider a neural field model where activities are observed only indirectly via a point-process, thus adressing the problem of inferring latent states. They infer both these states and the point-process parameters, but assume known parameters and neglect finite-size effects for the mesoscopic model; in contrast, here we inferred the mesoscopic model parameters while assuming that population states are observed. Inferring both mesoscopic model parameters and latent states remains a challenge for both of these approaches.

To obtain posteriors, we employed a Hamiltonian Monte Carlo algorithm with minimal automatic tuning. We found this to work better than a more automatically tuned variant (c.f. Section 4.4.6), but it is beyond the scope of this work to provide a complete survey of sampling methods. The applicability of more recently developed algorithms such as Riemann manifold Monte Carlo [126], sequential Monte Carlo [127] and nested sampling [80] would be worth exploring in future work. Variational methods such as that described by Kucukelbir et al. [128] are another alternative to estimating posteriors that do not require sampling at all. They generally scale to large parameter spaces but do not provide the asymptotic guarantees of MCMC and may artifically smooth the resulting posterior.

Important obstacles to using inference on complex models are the implementation and computational costs. Software tools developed for this work have helped limit the former, but the latter remains a challenge, with many of the figures shown requiring multiple days of computation on a personal workstation. While manageable for studying fixed networks, this would become an impediment for scaling to larger models, or tracking the evolution of parameter values by inferring them on successive time windows. For such tasks further work would be required to reduce the inference time, for example by investigating how large the integration time step for the mesoGIF model can be made, or by optimizing the current implementation. One might also attempt to derive a computationally simpler model, or make better use of parallelization and/or graphical processing units.

As noted by Rule et al. [125, §3.2], an additional complication to inferring mechanistic model parameters is that they may be under-constrained. In our case, since mesoGIF is a rate model, the voltage scale can be chosen freely by setting the resting (u_{rest}) and threshold (u_{th}) potentials—if we nonetheless attempt to infer them along with the noise scale (Δ_u), fits are unable to converge (see Sections 4.2.2 and 4.4.5). We avoided this problem by identifying the problematic parameters and fixing them to their known values. However, the development of a more systematic approach to dealing with under-constrained parameters is left for future investigations.

Since inference time is highly dependent on computational complexity, there is a trade-off between bottom-up models that attempt to match dynamics as closely as possible, and simpler top-down models that aim for computational efficiency: while the latter tend to provide better scalability, the former more precisely reproduce biological constraints, and thus better allow for extrapolation to new dynamical regimes (see Section 4.2.6). And while simpler models may allow interpretations based on correlations, only more accurate models can provide the kind of causal interpretation needed to go from *description* to *prediction*. Choosing the right model thus remains a key component of data analysis and modelling.

Inference methods based on machine learning allow for flexible model design, using known biophysical parameter values when they are available, and inference to determine the others such that outputs are consistent with data. We hope this work further motivates the use of richer models in neuroscience, by providing tools to fit and validate them.

4.4. Methods

4.4.1. Microscopic model

We consider an ensemble of neurons grouped into M populations; the symbols i, j are used to label neurons, and α, β to label populations. The neuron indices i, j run across populations and are thus unique to each neuron.

Each neuron i produces a spike train represented as a sum of Dirac delta functions,

$$s_i(t) = \sum_k \Delta(t - t_{i,k})\,, \tag{4.9}$$

where $t_{i,k}$ is the time of its k-th spike. We denote Γ_i^β the set of neuron indices from population β that are presynaptic to neuron i, $w_{\alpha\beta}$ the strength of the connection from a neuron in population β to another in population α, and $\Delta_{\alpha\beta}$ the transmission delay between the two populations. As in [20], we assume that intrinsic neural parameters are homogeneous across a given population. We further assume that connection strengths depend only on the source and target populations; for a connection between neurons of population β to those of population α , the strength is either $w_{\alpha\beta}$ with probability $p_{\alpha\beta}$ or zero with probability $1 - p_{\alpha\beta}$. Each spike elicits a post-synaptic current, which we sum linearly to obtain the synaptic inputs to neuron i from M populations,

$$R_\alpha I_{\text{syn},i}(t) = \tau_m^\alpha \sum_{\beta=1}^{M} w_{\alpha\beta} \sum_{j\in\Gamma_i^\beta} \big(\epsilon_{\alpha\beta} * s_j\big)(t)\,. \tag{4.10}$$

The transmission delay is captured by shifting the synaptic kernel with a Heaviside function θ:

$$\epsilon_{\alpha\beta}(t) = \theta(t - \Delta_{\alpha\beta}) \frac{e^{-(t-\Delta)/\tau_{s,\beta}}}{\tau_{s,\beta}}\,. \tag{4.11}$$

Spike generation is modeled by a *generalized integrate-and-fire* mechanism: leaky integration with adapting threshold, followed by an escape rate process. For each neuron i, the membrane potential u_i and firing threshold ϑ_i evolve

Table 4.2.: Parameters for the micro- and mesoscopic models. For the mesoscopic populations, the ensemble of neuron parameters is replaced by a single effective value for that population. For each parameter, we indicate the number of components in the two and four population models; adaptation parameters have fewer components because the model assumes no adaptation for inhibitory neurons. Boldface is used to indicate inferred parameters; the remainder are fixed to the known ground truth values listed in Table 4.7. This results in respectively 14 and 36 free parameters for the two- and four-population models. A brief discussion of how we chose which parameters to infer is given at the end of Section 4.4.5.

	No. of components		
Parameter	2 pop.	4 pop.	Description
p	4	16	connection probability
$\mathbf{w}$	**4**	**16**	**connection weight**
Δ	4	16	transmission delay
N	2	4	no. of neurons in pop.
R	2	4	membrane resistance
u_{rest}	2	4	membrane resting potential
$\boldsymbol{\tau}_{\mathbf{m}}$	**2**	**4**	**membrane time constant**
t_{ref}	2	4	absolute refractory period
u_{th}	2	4	non-adapting threshold
u_{r}	2	4	reset potential
$\mathbf{c}$	**2**	**4**	**escape rate at threshold**
Δ_{u}	**2**	**4**	**noise level**
$\boldsymbol{\tau}_{\mathbf{s}}$	**2**	**4**	**synaptic time constant**
$\mathbf{J}_{\theta}$	**1**	**2**	**adaptation strength**
$\boldsymbol{\tau}_{\theta}$	**1**	**2**	**adpatation time constant**

according to

$$\tau_{m,\alpha}\frac{du_i}{dt} = -u_i + u_{\text{rest},\alpha} + R_\alpha I_{\text{ext},\alpha}(t) + R_\alpha I_{\text{syn},i}(t)\,; \tag{4.12}$$

$$\vartheta_i(t) = u_{\text{th},\alpha} + \int_{-\infty}^{t} \theta_\alpha(t-t')s_i(t')\,\mathrm{d}t'\,. \tag{4.13}$$

Here, θ_α is the adaptation kernel for population α and $I_{\text{ext},\alpha}$ the external input to that population. For this work we used an exponential adaptation kernel,

$$\theta_\alpha(t) = \frac{J_{\theta,\alpha}}{\tau_{\theta,\alpha}} e^{-t/\tau_\theta}\,, \tag{4.14}$$

which allows us to rewrite Equation (4.13) as

$$\tau_\theta \frac{d\vartheta_i}{dt}(t) = -\vartheta_i(t) + u_{\text{th},\alpha} + J_{\theta,\alpha}s_i(t)\,. \tag{4.15}$$

Spikes are generated stochastically with an escape rate (also called conditional intensity or hazard rate), calculated with the inverse link function f:

$$\Lambda_i(t) = f(u_i(t) - \vartheta_i(t))\,. \tag{4.16}$$

For this work we used

$$\Lambda_i(t) = c_\alpha \exp((u_i(t) - \vartheta_i(t))/\Delta_{\text{u},\alpha})\,, \tag{4.17}$$

where Δ_{u} parameterizes the amount of noise (or equivalently, the softness of the threshold) and c is the firing rate when $u(t) = \vartheta(t)$.

Once a spike is emitted, a neuron's potential is reset to u_r and clamped to this value for a time t_{ref} corresponding to its absolute refractory period. It then evolves again according to Equation (4.12). All model parameters are summarized in Table 4.2.

4.4.2. Mesoscopic model

The mesoscopic equations describe the interaction of population activities (total number of spikes per second per neuron) in closed form: they can be integrated without the need to simulate indiviual neurons. This is achieved by identifying each neuron i by its age τ_i and making the assumptions stated in Section 4.2.1: that each population is homogeneous, that neurons are all-to-all connected with effective weights $p_{\alpha\beta}w_{\alpha\beta}$, and that dynamics are well approximated as a quasi-renewal process. Under these conditions it is possible to rewrite the dynamical equations in terms of the refractory densities $\rho_\alpha(t,\tau)$ – the proportion of neurons with age $\tau_i \in [\tau, \tau + d\tau)$ in each population α. With very large populations N_α, we can neglect finite-size fluctuations and ρ satisfies the transport equation [8, 40, 106, 129]:

$$\frac{\partial \rho_\alpha}{\partial t} + \frac{\partial \rho_\alpha}{\partial \tau} = -\Lambda_\alpha(t,\tau)\,\rho_\alpha\,, \qquad \rho_\alpha(t,0) = A_\alpha(t)\,. \tag{4.18}$$

Neuronal dynamics and synaptic interactions are captured within the functional form of the hazard rate $\Lambda_\alpha(t,\tau)$, which depends only on τ and on the history of population activities. In the limit $N_\alpha \to \infty$, the evolution of $A(t)$ matches its expectation $a(t)$ and is obtained by integrating over all neurons:

$$N_\alpha \to \infty : \quad A_\alpha(t) = a_\alpha(t) = \int_0^\infty \Lambda_\alpha(t,\tau)\rho_\alpha(t,\tau)\,\mathrm{d}\tau\,. \tag{4.19}$$

For finite N, the expression for the expected activity becomes [20, 45]

$$a_\alpha(t) = \int_0^\infty \Lambda_\alpha(t,\tau)\,\rho_\alpha(t,\tau)\,d\tau + \Lambda_\alpha(t)\left(1 - \int_0^\infty \rho_\alpha(t,\tau)\,d\tau\right), \tag{4.20}$$

where $\Lambda_\alpha(t)$ is a rate function that accounts for finite-size effects in the refractory density. The activity then follows a stochastic process described by

$$A_\alpha(t) = \frac{n_\alpha(t)}{N dt}\,, \qquad n_\alpha(t) \sim \mathrm{Binom}(N\,a(t)\,dt\,;\ N_\alpha)\,. \tag{4.21}$$

Intuition for Equation (4.20)

Note that in the finite-size case, noise is added to Equation (4.18), such that ρ_α no longer remains normalized: if we only integrated over ρ_α as in Equation (4.19), we would under- or over-count neurons. This is corrected for with the second term in Equation (4.20): the part in parenthesis corresponds to the number of under- or over-counted neurons.

For this work we discretize time into steps of length Δt, and instead of the refractory density work with the vector $m_\alpha^{(k)}$, where $m_{\alpha,l}^{(k)}$ is formally defined as the expected number of neurons of age $\tau \in [l\Delta t, (l+1))$:

$$m_{\alpha,l}^{(k)} = \int_\tau^{\tau+\Delta t^-} N_\alpha \rho_\alpha(t_k, l\,\Delta t)\,d\tau\,, \qquad (l = 1, \dots, K < \infty)\,. \tag{4.22}$$

Here the superscript (k) indicates the simulation time step and l the age bin. Since refractory effects are negligible for sufficiently old neurons, $m^{(k)}$ only needs to be computed for a finite number of age bins K (see Appendix 4.E, as well as Equation (86) from Schwalger, Deger, and Gerstner [20]).

We similarly compute the firing rates at time t_k as a vector $\Lambda_{\alpha,l}^{(k)}$, $l = 1, \dots, K$. The expected number of spikes in a time bin,

$$\bar{n}_\alpha^{(k)} = \mathrm{E}\left[n_\alpha^{(k)}\right], \tag{4.23}$$

can then be computed in analogy with Equation (4.20), by summing the products $\Lambda_{\alpha,l}^{(k)} m_{\alpha,l}^{(k)}$ over l and adding a finite-size correction; the precise equations used to evaluate $m_{\alpha,l}^{(k)}$, $\Lambda_{\alpha,l}^{(k)}$ and $\bar{n}_\alpha^{(k)}$ are listed in Appendix 4.E. We can convert spike counts to activities by dividing by $N_\alpha \Delta t$:

$$a_\alpha^{(k)} := \frac{\bar{n}_\alpha^{(k)}}{N_\alpha \Delta t}\,, \qquad A_\alpha^{(k)} := \frac{n_\alpha^{(k)}}{N_\alpha \Delta t}\,. \tag{4.24}$$

For the following, it will be convenient to define the single-neuron firing probability,

$$p_{\alpha,\eta}^{(k)} \coloneqq \frac{\tilde{n}_{\alpha,\eta}^{(k)}}{N_\alpha}\,, \tag{4.25}$$

where the subscript η makes explicit the dependence on the model parameters. This allows us to rewrite Equation (4.4) as

$$n_\alpha^{(k)} \sim \mathrm{Binom}\left(p_{\alpha,\eta}^{(k)}; N_\alpha\right)\,, \tag{4.26}$$

where $p_{\alpha,\eta}^{(k)} = p_{\alpha,\eta}(t_k|\mathcal{H}_{t_k})$ depends on the activity history $\mathcal{H}_{t_k}$ (Equation (4.7)). Because K is finite, we can replace $\mathcal{H}_{t_k}$ by a finite state-vector $S^{(k)}$, obtained by concatenating all variables required to update $n^{(k)}$ (see Appendix 4.E, especially Equation (4.42)):

$$S^{(k)} = \left(n^{(k)}, m^{(k)}, \Lambda^{(k)}, \dots\right)\,. \tag{4.27}$$

The update equations for $S^{(k)}$ are Markovian by construction, which simplifies the expression of the model's likelihood presented in the next section.

4.4.3. Likelihood for the mesoscopic model

As stated in Section 4.4.2, the mesoGIF model can be cast in a Markovian form, which allows us to expand the probability of observing a sequence of spike counts as a recursive product. If that sequence has length L and an initial time point k_0, then its probability is

$$p\left(\left\{n_\alpha^{(k)}\right\}_{\substack{k=k_0\dots k_0+L-1\\ \alpha=1\dots M}}\right) = \prod_{\alpha=1}^{M}\prod_{k=k_0}^{L+k_0-1} p\left(n_\alpha^{(k)} \mid S^{(k)}\right)\,. \tag{4.28}$$

The likelihood of this sequence then follows directly from the probability mass function of a binomial, using the definitions for $n_\alpha^{(k)}$ and $p_{\alpha,\eta}^{(k)}$ defined above;

$$L_{k_0;L} = \prod_{\alpha=1}^{M}\prod_{k=k_0}^{k_0+L-1} \binom{N_\alpha}{n_\alpha^{(k)}} \left(p_{\alpha,\eta}^{(k)}\right)^{n_\alpha^{(k)}} \left(1-p_{\alpha,\eta}^{(k)}\right)^{N_\alpha - n_\alpha^{(k)}}\,. \tag{4.29}$$

We note that the $n_\alpha^{(k)}$ are observed data points, and are thus constant when maximizing the likelihood.

Expanding the binomial coefficient, the log-likelihood becomes

$$\begin{aligned}\log L_{k_0;L}(\eta) = \sum_{\alpha=1}^{M}\sum_{k=k_0}^{k_0+L-1} &\log\left(N_\alpha!\right) - \log\left(n_\alpha^{(k)}!\right) - \log\left((N_\alpha - n_\alpha^{(k)})!\right) \\ &+ n_\alpha^{(k)}\log\left(\tilde{p}_{\alpha,\eta}^{(k)}\right) + \left(N_\alpha - n_\alpha^{(k)}\right)\log\left(1-\tilde{p}_{\alpha,\eta}^{(k)}\right)\,,\end{aligned} \tag{4.30}$$

where we clipped the probability $\tilde{p}_\alpha^{(k)}$ to avoid writing separate expressions for $p_{\alpha,\eta}^{(k)} \in 0, 1$,

$$\tilde{p}_{\alpha,\eta}^{(k)} = \begin{cases} \epsilon & \text{if } p_{\alpha,\eta}^{(k)} \le \epsilon\,, \\ p_\alpha^{(k)} & \text{if } \epsilon \le p_{\alpha,\eta}^{(k)} \le 1-\epsilon\,, \\ 1-\epsilon & \text{if } p_{\alpha,\eta}^{(k)} \ge 1-\epsilon\,. \end{cases} \tag{4.31}$$

Clipping also avoids issues where the firing probability $p_\alpha^{(k)}$ exceeds 1, which occurs when one explores the parameter space. (This can happen when parameters are such that the chosen Δt is no longer small enough for the underlying Poisson assumption to be valid, although it should *not* occur around the true parameters. See the discussion by

fit parameter	value	comment
learning rate	0.01	Adam parameter
β_1	0.1	Adam parameter
β_2	0.001	Adam parameter
g_{clip}	100	clipping threshold
L_{burnin}	10 s	data burn-in
B_{burnin}	0.3 s	mini-batch burn-in
Γ_L	1	L_{burnin} noise factor
Γ_B	0.1	B_{burnin} noise factor

Table 4.3.: Fitting parameters for Adam. Learning rate, β_1 and β_2 are as defined in Kingma and Ba [68].

Schwalger, Deger, and Gerstner [20, p. 48].) We found that with double precision, a tolerance $\epsilon = 1 \times 10^{-8}$ worked well.

For numerical stability, logarithms of factorials are computed with a dedicated function such as SciPy's `gammaln` [130]. For optimization, the term $\log(N_\alpha!)$ can be omitted from the sum since it is constant.

4.4.4. Initializing the model

Although the updates to the state S are deterministic (see Section 4.4.2), only the components $n_\alpha^{(k_0)}$ of the initial state $S^{(k_0)}$ is known – unobserved components can easily number in the thousands. We get around this problem in the same manner as in Schwalger, Deger, and Gerstner [20]: by making an initial guess that is consistent with model assumptions (survival counts sum to N_α, etc.) and letting the system evolve until it has forgotten its initial condition. We note that the same problem is encountered when training recurrent neural networks, whereby the first data points are used to "burn-in" unit activations before training can begin. For the results we presented, we used a variation of the initialization scheme used by Schwalger, Deger, and Gerstner [20] that we call the "silent initialization".

Silent initialization Neurons are assumed to have never fired, and thus they are all "free". This results in large spiking activity in the first few time bins, which then relaxes to realistic levels.

Algorithm 1 Silent initialization scheme.

1: $n_\alpha \leftarrow 0$
2: $h_\alpha, u_{\alpha,i} \leftarrow u_{\text{rest},i}$
3: $x_\alpha \leftarrow N_\alpha$
4: $\Lambda_{\alpha,i}, \Lambda_{\text{free},\alpha}, g_\alpha, m_{\alpha,i}, v_{\alpha,i}, y_{\alpha\beta}, z_\alpha \leftarrow 0$

This initialization scheme has the advantage of being simple and needing no extra computation, but with the high-dimensional internal state S, also requires a large burn-in time of around 10 s. This can be largely mitigated by using sequential batches (Algorithm 2).

We also experimented with intializing the model at a stationary point (Appendix 4.C), but in the cases we considered it did not provide a notable improvement in computation time.

4.4.5. Estimating parameters

To maximize the likelihood, we used Adam [68], a momentum-based stochastic gradient descent algorithm, for which gradients were computed automatically with Theano [131] (see Section 4.4.9). Training parameters are listed in Table 4.3.

Despite the similarities, there remain important practical differences between fitting the mesoscopic model and training a recurrent neural network (RNN). Notably, RNN weights are more freely rescaled, allowing the use of single precision floating point arithmetic. In the case of the mesoscopic model, the dynamic range is wider and we found it necessary to use double precision.

Compared to a neural network, the mesoscopic update equations (Equations (4.43–4.64)) are also more expensive to compute, in our case slowing down parameter updates by at least an order of magnitude.

The subsequences of data ("mini-batches") used to train an RNN are usually selected at random: at each iteration, a random time step k_0 is selected, from which the next B_{burnin} data points are used for burn-in and the following B data points form the mini-batch. This becomes problematic when long burn-in times are required, not only because it requires long computation times, but also because it wastes a lot of data. We addressed this problem by keeping the state across iterations (Alg. 2): since this is a good guess of what it should be after updating the parameters, it reduces the required burn-in time by an order of magnitude. However this requires batches to follow one another, breaking the usual assumption that they are independently selected. In practice this seemed not to be a problem; in anecdotal comparisons, we found that training with either a) randomly selected batches and stationary initialization (Algorithm 3), or b) sequential batches and silent initialization (Algorithm 1), required comparable numbers of iterations to converge to similar parameter values. Computation time in the case of random batches however was much longer.

We also found that bounding the gradient helped make inference more robust. We set maximum values for each gradient component and rescaled the gradient so that no component exceeded its maximum (Alg. 2, lines 7 to 10).

Maximizing the posterior rather than the likelihood by multiplying the latter by parameter priors (to obtain the MAP estimate rather than the MLE) helped prevent the fit from getting stuck in unphysical regions far from the true parameters, where the likelihood may not be informative. We used noninformative priors (see Table 4.7) so as to ensure that they didn't artificially constrain the fits. Fits were also initialized by sampling from the prior.

Choosing adequate external inputs may also impact fit performance, as in general, sharp stimuli exciting transients on multiple timescales tend to be more informative than constant input [132]. That being said, even under constant input, the fluctuations in a finite-sized neuron population still carry some information, and anecdotal evidence suggests that these can be sufficient to infer approximate model parameters. In this paper, we used a sinusoidal input with frozen white noise to train the mesoGIF model—with only one dominant time scale, this input is more informative than constant input but far from optimal for the purpose of fitting. This made it a reasonable choice for computing baseline performance measures.

Finally, to allow fits to converge, it is essential to avoid fitting any ill-defined or degenerate parameters. For example, as explained in Section 4.2.2, we fixed the parameters u_{rest} and u_{th} because the mesoGIF model is invariant under a rescaling of the voltage; for simplicity we also fixed u_{r} and R even though this was not strictly necessary. The parameters w and p are similarly degenerate (see Equation (4.48)) and we fixed p. The parameters N, Δ and t_{ref} are effectively discrete (either in numbers of neurons or time bins), and they were also fixed to simplify the implementation. Table 4.2 summarizes the inferred and non-inferred parameters.

4.4.6. Estimating the posterior

The posteriors in Section 4.2.5 were obtained using Hamiltonian Monte Carlo [63, 117] (HMC). Having expressed the likelihood with Theano made it straightforward to use the implementation in PyMC3 [61]— HamiltonianMC, to sample the likelihood; the sampling parameters we used are listed in Table 4.4.

Although straightforward, this approach pushes the limit of what can be achieved with currently implemented samplers: because the likelihood of this model is expensive to evaluate, even coarse distributions can take hours to obtain. In addition, the large state vector required sufficiently large amounts of memory to make the automatically tuned NUTS [133] sampler impractical. (NUTS stores a theoretically unbounded tree of the most recent states in order to tune the sampling parameters.) In an application with experimental data, one would want to reserve sufficient computational resources to perform at least basic validation of the obtained that posterior, using for example the methods described by Gelman et al. [134] and Talts et al. [135].

Algorithm 2 Training with **sequential mini-batches**. The gradient is normalized before computing Adam updates. Note that the state is not reinitialized within the inner loop.

repeat
  $S \leftarrow$ initialize state
  $k' \sim \mathrm{Uniform}(0, \Gamma_L B)$ ▷ Randomize initialization burn-in
  $k_0 \leftarrow L_{\mathrm{burnin}} + k'$
  while $k_0 < L - B$ **do** ▷ Scan data sequentially
    $g \leftarrow \nabla \log L(\eta, A_{k_0 : k_0+B})$ ▷ Log-likelihood gradient on the mini-batch
    if $\mathrm{any}(|g| > g_{\mathrm{clip}})$ **then** ▷ Normalize gradients with L^∞ norm
      $g_{\max} \leftarrow \max(|\Delta\eta|)$
      $g \leftarrow \frac{g_{\mathrm{clip}}}{g_{\max}} \Delta\eta$
    end if
    $\eta \leftarrow Adam(g)$ ▷ Update parameters updates with Adam
    $k' \sim \mathrm{Uniform}(B_{\mathrm{burnin}}, (1 + \Gamma_B) B_{\mathrm{burnin}})$ ▷ Randomize batch burn-in
    $k_0 \leftarrow k_0 + k'$
  end while
until converged.

Algorithm	`HamiltonianMC` (PyMC3[61])
step scale	0.0025
path length	0.1
tuning steps	20
initialization	`jitter+adapt_diag`
start	η_{MAP} estimate
no. of samples	2000
total run time	201 h

Table 4.4.: Specification of the MCMC sampler.

In order for samplers to find the high probability density region in finite time, we found it necessary to initialize them with the MAP estimate. This also ensured that their mass matrix was tuned on an area of the posterior with appropriate curvature. In applications where the posterior has multiple modes, one should be able to identify them from the collection of fits. The high probability density region around each mode should then be sampled separately, integrated, and combined with the others to obtain the full posterior. (See e.g. van Haasteren [136] for integration methods for MCMC chains.)

Finally, as with parameter optimization, we found that the use of at least double precision floats was required in order to obtain consistent results.

4.4.7. Measuring performance

In order to assess the performance of our inference method, we quantified the discrepancy between a simulation using ground truth parameters and another using inferred parameters; the same input was used for both simulations, and was different from the one used for training. Following Augustin et al. [116], discrepancy was quantified using both correlation (ρ) and root mean square error (RMSE); these are reported according to the amount of data L used to train the model, which may be given either in time bins or seconds.

The correlation between activity traces from the ground truth and inferred models, respectively $A^{\mathrm{true}}(t)$ and $\hat{A}^{(L)}(t)$, was obtained by computing the *per-trial* Pearson coefficient for each of the M populations and averaging the results

across populations to report a single value:

$$\rho(A^{\text{true}}, \hat{A}^{(L)}) = \frac{1}{M}\sum_{\alpha=1}^{M} \frac{\left\langle \left(A_\alpha^{\text{true}} - \langle A_\alpha^{\text{true}}\rangle\right)\left(\hat{A}_\alpha^{(L)} - \langle \hat{A}_\alpha^{(L)}\rangle_k\right)\right\rangle_k}{\sqrt{\left\langle \left(A_\alpha^{\text{true}} - \langle A_\alpha^{\text{true}}\rangle_k\right)^2 \left(\hat{A}_\alpha^{(L)} - \langle \hat{A}_\alpha^{(L)}\rangle_k\right)^2\right\rangle_k}}\,. \tag{4.32}$$

Here brackets indicate averages over time:

$$\langle A\rangle_k := \frac{1}{L'}\sum_{k=k_0}^{k_0+L'} A^{(k)}\,,$$

with k a discretized time index. The initial time point k_0 sets the burn-in period; in all calculations below, we set it to correspond to 10 s to ensure that any artifacts due to the initialization have washed away. The value of L' need not be the same as L, and we set it to 9000 (corresponding to 9 s) for all discrepancy estimates.

As with correlation, the per-trial RMSE was averaged across populations,

$$\text{RMSE}(A^{\text{true}}, \hat{A}^{(L)}) := \sqrt{\frac{1}{M}\sum_{\alpha=1}^{M}\left\langle \left(\hat{A}_\alpha^{(L)} - A_\alpha^{\text{true}}\right)^2\right\rangle_k}\,. \tag{4.33}$$

Because the models are stochastic, Equations (4.32) and (4.33) describe random variables. Thus, for each of our results, we generated ensembles of realizations $\{A^{\text{true},r}\}_{r=1}^{R_1}$, $\{A^{\text{true},r'}\}_{r=1}^{R_2}$ and $\{\hat{A}^r\}_{r=1}^{R_3}$, each with a different set of random seeds. We compute the ρ and RMSE for the $R_1 \times R_2$ pairs $(A^{\text{true},r}, \hat{A}^r)$, as well the $R_1 \times R_3$ combinations $(A^{\text{true},r}, A^{\text{true},r'})$, from which we empirically estimate the mean and standard deviation of those measures. Values for the pairs $(A^{\text{true},r}, A^{\text{true},r'})$ provide an estimate of the best achievable value for a given measure.

Another way to address the stochasticity of these measures is to use *trial-averaged* traces:

$$\bar{\rho}(L) = \rho(\bar{A}^{\text{true}}, \hat{\bar{A}})\,, \tag{4.34}$$

$$\overline{\text{RMSE}}(L) = \text{RMSE}(\bar{A}^{\text{true}}, \hat{\bar{A}})\,; \tag{4.35}$$

where the trial-averaged activity,

$$\bar{A}_\alpha^{(k)} := \frac{1}{R}\sum_{r=1}^{R} A_\alpha(t_k \mid \mathcal{H}_{t_k}^r)\,,$$

is as in Equation (4.5). Because trial-averaged measures only provide a point estimate, we used bootstrapping to estimate their variability. We resampled the ensemble of realizations with replacement to generate a new ensemble of same size R, and repeated this procedure 100 times. This yielded a set of R measures (either $\bar{\rho}$ or $\overline{\text{RMSE}}$), for which we computed the sample standard deviation. Note that in contrast to per-trial measures, errors on trial-averaged measurements vanish in the limit of large number of trials R and thus are not indicative of the variability between traces.

We found the pair of measures $(\bar{\rho}, \text{RMSE})$ (Equations (4.33) and (4.34)) to provide a good balance between information and conciseness (c.f. Section 4.2.3). We generally used $R_1 = R_2 = 50$ and $R_3 = 100$ for the ensembles, with the exception of Figure 4.4 where $R_1 = R_2 = R_3 = 20$. We also ensured that sets of trial-averaged measures use the same number of trials, to ensure comparability.

4.4.8. Stimulation and integration details

All external inputs used in this paper are shared within populations and frozen across realizations. They are distinct from the escape noise (Equations (4.1) and (4.4)), which is *not* frozen across realizations.

	2 pop. model		4 pop. model				Unit
	E	I	L2/3e	L2/3i	L4e	L4i	
B	0.25	0.1	0.0	0.0	0.25	0.1	mA
ω	2.0	2.0	2.0	2.0	2.0	2.0	–
q	4.0	4.0	4.0	4.0	4.0	4.0	mA

Table 4.5.: Parameters for the sine-modulated input.

Sine-modulated white noise input For inferring parameters in all our work, we generated training data with a sine-modulated stimulus of the form

$$I_{\text{ext}}(t) = B\sin(\omega t)\cdot(1 + q\xi(t))\,, \tag{4.36}$$

where $\xi(t)$ is the output of a white noise process with $\langle\xi(t)\xi(t')\rangle = \Delta(t-t')$. This input was chosen to be weakly informative, in order to provide a baseline for the inference procedure. The values of B, ω and q are listed in Table 4.5. The integration time step was set to 0.2 ms for microscopic simulations and 1 ms for mesoscopic simulations. We then tested the fitted model with the inputs described below.

OU process input Fit performance in Sections 4.2.3 and 4.2.6 was measured using an input produced by an Ornstein-Uhlenbeck (OU) process defined by

$$\frac{dI_{\text{test}}}{dt} = -\frac{(I_{\text{test}} - \mu_{\text{OU}})}{\tau_{\text{OU}}}\,dt + \sqrt{\frac{2}{\tau_{\text{OU}}}}q\,dW\,. \tag{4.37}$$

Here μ_{OU}, τ_{OU} and q respectively set the mean, correlation time and noise amplitude of the input, while dW denotes increments of a Wiener process. The parameter values and initial condition ($I_{\text{test}}(0)$) are listed in Table 4.6.

	2 pop. model		4 pop. model				unit
	E	I	L2/3e	L2/3i	L4e	L4i	
μ_{OU}	0.1	0.05	1	1	0	1	mA
τ_{OU}	1	1	2	2	–	2	s
q	0.125	0.125	0.5	0.5	0	0.5	mA
$I_{\text{test}}(0)$	0.1	0.05	0.5	0.5	0	0.5	mA

Table 4.6.: Parameters for the OU-process input (Equation (4.37)).

Impulse input We further tested the generalizability of the four population model using an input composed of sharp synchronous ramps. As the transient response is qualitatively different from the sinusoidal oscillations used to fit the model, this is a way of testing the robustness of the inferred parameters to extrapolation. The input had the following form:

$$I_\alpha(t) = \sum_{t_0\in\mathcal{T}} \mathcal{J}_{t_0}(t) \tag{4.38}$$

$$\mathcal{J}_{t_0}(t) = \begin{cases} B\left(1 - \frac{|t-t_0|}{d}\right) & \text{if } |t-t_0| \le d, \\ 0 & \text{otherwise,} \end{cases} \tag{4.39}$$

and was generated with $d = 0.15$ s, $B = (0, 0, 0.6, -0.6)$ mA. Impulses were placed at

$$\mathcal{T} = \{11.0, 11.7, 12.2, 12.9, 14.1, 14.5, 15.5, 15.8, 16.2, 16.8\}\,\text{s}\,.$$

Numerical integration For all simulations of the mesoGIF model, we used a time step of 1 ms. We also used a 1 ms time step when inferring parameters. Simulations of the microscopic GIF model require finer temporal

resolution, and for those we used time steps of 0.2 ms. In order to have the same inputs at both temporal resolutions, they were generated using the finer time step, and coarse-grained by averaging.

We used the Euler-Maruyama scheme to integrate inputs; the GIF and mesoGIF models are given as update equations of the form $A(t + \Delta t) = F(A(t))$, and thus already define an integration scheme.

4.4.9. Software

We developed software for expressing likelihoods of dynamical systems by building on general purpose machine learning libraries: `Theano_shim` (`https://github.com/mackelab/theano_shim`) is a thin layer over the numerical backend, allowing one to execute the same code either using Theano [131] or Numpy [130]. `Sinn` (`https://github.com/mackelab/sinn`) makes use of `theano_shim` to provide a backend-agnostic set of high-level abstractions to build dynamical models. Finally, a separate repository (`https://github.com/mackelab/fsGIF`) provides the code specific to this paper.

Acknowledgments

We thank Pedro Gonçalves, Giacomo Bassetto, Tilo Schwalger and David Dahmen for discussions and comments on the manuscript. AR and AL were supported by NSERC (Canada); AL also acknowledges support from the Humboldt Foundation. JHM was supported by the German Research Foundation (DFG) through SFB 1089, and the German Federal Ministry of Education and Research (BMBF, project 'ADMIMEM', FKZ 01IS18052 A-D).

Appendix

4.A. Priors and parameter values

For both microscopic and mesoscopic models, unless otherwise specified in the text, we used the same parameter values as our ground truth values. Values are listed in Table 4.7 and are based on those given in Schwalger, Deger, and Gerstner [20], and we follow the recommendation therein of adjusting resting potentials u_{rest} to maintain realistic firing rates. To facilitate simulations, we also reduced the population sizes by a factor of 50 and correspondingly up-scaled the connectivity weights by a factor of $\sqrt{50}$, to maintain a balanced E-I network [137].

Prior distributions on inferred parameters were set sufficiently broad to be considered noninformative. Prior distributions are independent of the population, so as to ensure that any inferred feature (e.g. excitatory vs inhibitory connections) is due to the data.

The two population heteregeneous model was obtained by sampling similar but tighter distributions as the prior (Table 4.8). Only membrane and adaptation time constants were sampled; other parameters were as in Table 4.7.

Table 4.7.: Default parameter values and priors; symbols are the same as in Table 4.2. Most values are those given in Tables 1 and 2 of Schwalger, Deger, and Gerstner [20]. Priors are given as scalar distributions because they are the same for all components. The PDF of $\Gamma(\alpha, \theta)$ is $\frac{x^{\alpha-1}e^{-x/\theta}}{\theta^{\alpha}\Gamma(\alpha)}$.

	Value			
Parameter	2 pop. model	4 pop. model	unit	Prior distribution
pop. labels	E, I	L2/3e, L2/3i, L4e, L4i		
N	(438, 109)	(413, 116, 438, 109)		
R	(19, 11.964)	(0, 0, 19, 11.964)	ω	
u_{rest}	(20, 19.5)	(18, 18, 25, 20)	mV	
p	$\begin{pmatrix} 0.0497 & 0.1350 \\ 0.0794 & 0.1597 \end{pmatrix}$	$\begin{pmatrix} 0.1009 & 0.1689 & 0.0437 & 0.0818 \\ 0.1346 & 0.1371 & 0.0316 & 0.0515 \\ 0.0077 & 0.0059 & 0.0497 & 0.1350 \\ 0.0691 & 0.0029 & 0.0794 & 0.1597 \end{pmatrix}$		
w	$\begin{pmatrix} 2.482 & -4.964 \\ 1.245 & -4.964 \end{pmatrix}$	$\begin{pmatrix} 1.245 & -4.964 & 1.245 & -4.964 \\ 1.245 & -4.964 & 1.245 & -4.964 \\ 1.245 & -4.964 & 2.482 & -4.964 \\ 1.245 & -4.964 & 1.245 & -4.964 \end{pmatrix}$	mV	$w \sim \mathcal{N}\left(0, 4^2\right)$
τ_m	(0.01, 0.01)	(0.01, 0.01, 0.01, 0.01)	s	$\log_{10} \tau_m \sim \mathcal{N}(-2, 2^2)$
t_{ref}	(0.002, 0.002)	(0.002, 0.002, 0.002, 0.002)	s	
u_{th}	(15, 15)	(15, 15, 15, 15)	mV	$u_{\text{th}} \sim \mathcal{N}(15, 10^2)$
u_{r}	(0, 0)	(0, 0, 0, 0)	mV	$u_{\text{r}} \sim \mathcal{N}(0, 10^2)$
c	(10, 10)	(10, 10, 10, 10)	Hz	$c \sim \Gamma(2, 5)$
Δ_{u}	(5, 5)	(5, 5, 5, 5)	mV	$\Delta_{\text{u}} \sim \Gamma(3, 1.5)$
Δ	0.001	0.001	s	
τ_s	(0.003, 0.006)	(0.003, 0.006, 0.003, 0.006)	s	$\log_{10} \tau_s \sim \mathcal{N}(-3, 3^2)$
J_θ	(1.0, 0)	(1.0, 0, 1.0, 0)	mV	$J_\theta \sim \Gamma(2, 0.5)$
τ_θ	(1.0, –)	(1.0, –, 1.0, –)	s	$\log_{10} \tau_\theta \sim \mathcal{N}(-1, 5^2)$

Heterogeneous model parameter	Distribution parameter μ	σ
$\log_{10} \tau_{m,E}$	-1.6	0.5
$\log_{10} \tau_{m,I}$	-1.8	0.5
$\log_{10} \tau_{\theta,E}$	-0.7	0.5

Table 4.8.: Distribution parameters for the heterogeneous model. Each parameter was sampled from a log-normal distribution $\log_{10} \mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. No adaptation was modeled in the inhibitory population, so $\tau_{\theta,I}$ was not sampled.

4.B. Inferred parameters

Parameter	Inferred value	Average heterogeneous value	Unit
w	$\begin{pmatrix} 1.59 & -5.05 \\ 0.73 & -3.43 \end{pmatrix}$	$\begin{pmatrix} 2.482 & -4.964 \\ 1.245 & -4.964 \end{pmatrix}$	mV
τ_m	(0.011, 0.008)	(0.056, 0.046)	s
c	(5.05, 5.22)	(10, 10)	Hz
Δ_u	(5.09, 4.09)	(5, 5)	mV
τ_s	(0.0046, 0.0109)	(0.003, 0.006)	s
J_θ	(0.538, 0)	(1.0, 0)	mV
τ_θ	(0.131, –)	(0.380, –)	s

Table 4.9.: Inferred parameters for a heterogeneous population (Section 4.2.4); values are given in vector format, as (η_E, η_I). Corresponding average values for the heterogeneous microscopic model are given for comparison. (The heterogeneous model was homogeneous in all parameters except τ_m and τ_θ.)

Table 4.10.: Inferred values for the 4 population model. The values for the homogeneous microscopic model used in Figures 4.7 and 4.8 are listed on the right. Theory predicts these to be the best parameterization for the mesoscopic model, and thus should be recovered by maximizing the posterior (MAP values). Since L2/3 receives no external input in the training data, the inferred parameters for those populations are understandably further from theory.

	MAP				Theory			
	$L2/3e$	$L2/3i$	$L4e$	$L4i$	$L2/3e$	$L2/3i$	$L4e$	$L4i$
$w_{L2/3e\leftarrow\cdot}$	0.734	−5.629	1.546	−5.292	1.245	−4.964	1.245	−4.964
$w_{L2/3i\leftarrow\cdot}$	1.181	−5.406	1.419	−4.294	1.245	−4.964	1.245	−4.964
$w_{L4e\leftarrow\cdot}$	1.528	−0.637	2.058	−4.213	1.245	−4.964	2.482	−4.964
$w_{L4i\leftarrow\cdot}$	0.174	1.112	1.046	−3.994	1.245	−4.964	1.245	−4.964
τ_m	0.016	0.015	0.008	0.009	0.010	0.010	0.010	0.010
c	16.717	18.170	9.020	9.680	10.000	10.000	10.000	10.000
Δu	7.435	6.453	4.750	4.420	5.000	5.000	5.000	5.000
τ_s	0.001	0.006	0.002	0.009	0.003	0.006	0.003	0.006
J_θ	0.232	—	0.967	—	1.000	—	1.000	—
τ_θ	0.425	—	1.596	—	1.000	—	1.000	—

4.C. Alternative initialization scheme

Compared to the silent initialization (Section 4.4.4), the "stationary initialization" finds a more realistic initial state, which reduces the burn-in time required by about an order of magnitude. This makes it more practical when minibatches are selected at random, and we used this scheme to validate Algorithm 2 (described in Section 4.4.5). However in general we found the computational gain to be offset by the added cost of solving a self-consistent equation for each batch.

Stationary initialization Assuming zero external input, we find a self-consistent equation for the stationary activity A^* (see Appendix 4.I). After solving numerically for A^*, the other state variables are then easily computed.

Algorithm 3 Stationary initialization scheme.

1: $A^*_\alpha \leftarrow$ Solve Equation (4.69)
2: $n_\alpha \leftarrow A^*_\alpha N_\alpha \Delta t$
3: $h_\alpha, y_{\alpha\beta}, u_{\alpha,i}, g_\alpha, \Lambda_{\text{free},\alpha}, \Lambda_{\alpha,i}, x_\alpha, z_\alpha, m_{\alpha,i}, v_{\alpha,i} \leftarrow$ Evaluate Eqs. (4.47–4.63) with A^*_α

4.D. Data requirements for different parameter sets

In Section 4.2.3, we showed that less than 10*s* of data were sufficient to infer the parameters of the two-population mesoGIF model. Of course, the exact data requirements will depend on how many parameters we need to infer and which they are (e.g. w vs τ_m).

To explore this issue, we repeated the inference procedure for the parameter subsets listed in Table 4.11, performing 24 fits for each subset using different amounts of data. Subsets η_1 and η_2 parameterize respectively the connectivity and the adaptation, while η_3 is the full set used for Figure 4.4. A similar figure to Figure 4.4 with all three subsets is shown in Figure 4.9a.

Table 4.11.: Definition of parameter subsets for the two population model. There are only two adaptation parameters because inhibitory populations have no adaptation in this model.

Subset label	Included parameters
η_1	$\{w_{EE}, w_{EI}, w_{IE}, w_{II}\}$
η_2	$\{\tau_{\theta,E}, J_{\theta,E}\}$
η_3	$\eta_1 \cup \eta_2 \cup \{c_E, c_I, \Delta_{\text{u},E}, \Delta_{\text{u},I}, \tau_{m,E}, \tau_{m,I}, \tau_{s,E}, \tau_{s,I}\}$

With the smaller subsets (η_1, η_2), 1.25 s of data were sufficient to get good accuracy of the inferred dynamics (Figure 4.9a). However working with such small amounts of data incurs a substantial computational cost. Firstly because the fits converge less consistently, thus requiring more fits to find a good estimate of the MAP (Figure 4.9, b, c and d left). And secondly because the algorithm optimizations making use of the longer traces (see Section 4.4.5) are no longer as effective, making each iteration slower on average.

Since we know the ground truth parameters, we can further estimate the expected error by computing the relative difference between true and inferred parameter values. For a parameter η and its estimate $\eta^{(L)}$ obtained by using L seconds of data, this is calculated as

$$\Delta_{\text{rel}}\left(\hat{\eta}^{(L)}\right) := \left|\frac{\hat{\eta}^{(L)} - \eta}{\eta}\right|. \tag{4.40}$$

The number of fits required to achieve this performance will vary according to the nature and number of parameters; indeed with more parameters to infer, we found that fits terminated further from the true values. A simple way then to quantify the uncertainty of any one particular fit is the sample standard deviation σ_η of the set of found optima from a collection of fits. In order to make the σ_η comparable between parameters, we normalized by the parameter mean μ_η to obtain the coefficient of variation:

$$\left|CV(\eta^{(L)})\right| \stackrel{\text{def}}{=} \left|\sigma_{\eta^{(L)}}/\mu_{\eta^{(L)}}\right| \tag{4.41}$$

Relative error and CV values for all parameter subsets are listed in Tables 4.12 and 4.13.

Table 4.12.: Relative error for the fits shown in Section 4.2.3.

		L					
Subset	Parameter	1.25	2.00	3.00	5.00	7.00	9.00
η_1	w_{EE}	0.047	0.023	0.045	0.034	0.029	0.027
	w_{EI}	0.040	0.018	0.046	0.033	0.035	0.032
	w_{IE}	0.013	0.038	0.000	0.001	0.005	0.024
	w_{II}	0.018	0.005	0.022	0.018	0.012	0.005
η_2	$J_{\theta,E}$	0.002	0.000	0.011	0.004	0.010	0.009
	$\tau_{\theta,E}$	0.283	0.370	0.009	0.108	0.030	0.045
η_3	w_{EE}	0.345	0.348	0.151	0.001	0.084	0.067
	w_{EI}	0.238	0.043	0.079	0.132	0.067	0.072
	w_{IE}	0.017	0.556	0.630	0.427	0.244	0.178
	w_{II}	0.070	0.495	0.503	0.326	0.180	0.136
	$J_{\theta,E}$	0.016	0.094	0.369	0.385	0.092	0.016
	$\tau_{\theta,E}$	0.267	0.054	0.450	0.586	0.248	0.213
	c_E	0.420	0.376	0.469	0.382	0.239	0.160
	c_I	2.825	0.006	0.133	0.142	0.128	0.161
	Δu_E	0.215	0.182	0.090	0.086	0.092	0.052
	Δu_I	0.520	0.338	0.466	0.302	0.129	0.058
	$\tau_{m,E}$	0.190	0.117	0.057	0.177	0.052	0.037
	$\tau_{m,I}$	2.590	0.619	0.430	0.393	0.235	0.219
	$\tau_{s,E}$	0.744	0.101	0.038	0.101	0.039	0.142
	$\tau_{s,I}$	0.271	0.119	0.138	0.132	0.081	0.081

Table 4.13.: Coefficients of variation for the collections of fits shown in Section 4.2.3.

		L					
Subset	Parameter	1.25	2.00	3.00	5.00	7.00	9.00
η_1	w_{EE}	1.33	1.05	0.95	0.72	0.61	0.82
	w_{EI}	0.69	0.63	0.43	0.62	0.40	0.48
	w_{IE}	1.38	1.48	1.53	1.75	1.90	1.36
	w_{II}	0.40	0.46	0.56	0.39	0.61	0.65
η_2	$J_{\theta,E}$	2.56	1.49	1.50	1.13	1.13	1.42
	$\tau_{\theta,E}$	7.70	10.33	8.75	11.11	5.64	6.82
η_3	w_{EE}	31.63	31.73	29.26	37.84	38.29	31.49
	w_{EI}	183.77	127.48	30.84	88.90	44.35	85.80
	w_{IE}	64.08	100.98	1833.28	1143.47	1489.30	2405.89
	w_{II}	53.90	65.92	36.25	78.14	37.70	55.10
	$J_{\theta,E}$	94.03	82.51	30.99	59.25	40.58	53.40
	$\tau_{\theta,E}$	70.53	329.84	255.21	209.10	282.68	160.36
	c_E	83.08	65.81	37.16	39.42	48.50	47.80
	c_I	29.31	28.84	34.10	52.49	49.66	52.50
	Δu_E	14.98	19.16	6.11	11.37	7.58	8.84
	Δu_I	32.84	30.33	13.28	41.22	21.86	34.30
	$\tau_{m,E}$	429.43	420.71	428.10	431.68	441.05	444.64
	$\tau_{m,I}$	256.36	269.36	346.42	337.09	314.64	288.72
	$\tau_{s,E}$	427.66	441.61	447.11	446.85	446.84	447.20
	$\tau_{s,I}$	238.26	240.74	65.92	262.20	71.13	295.76

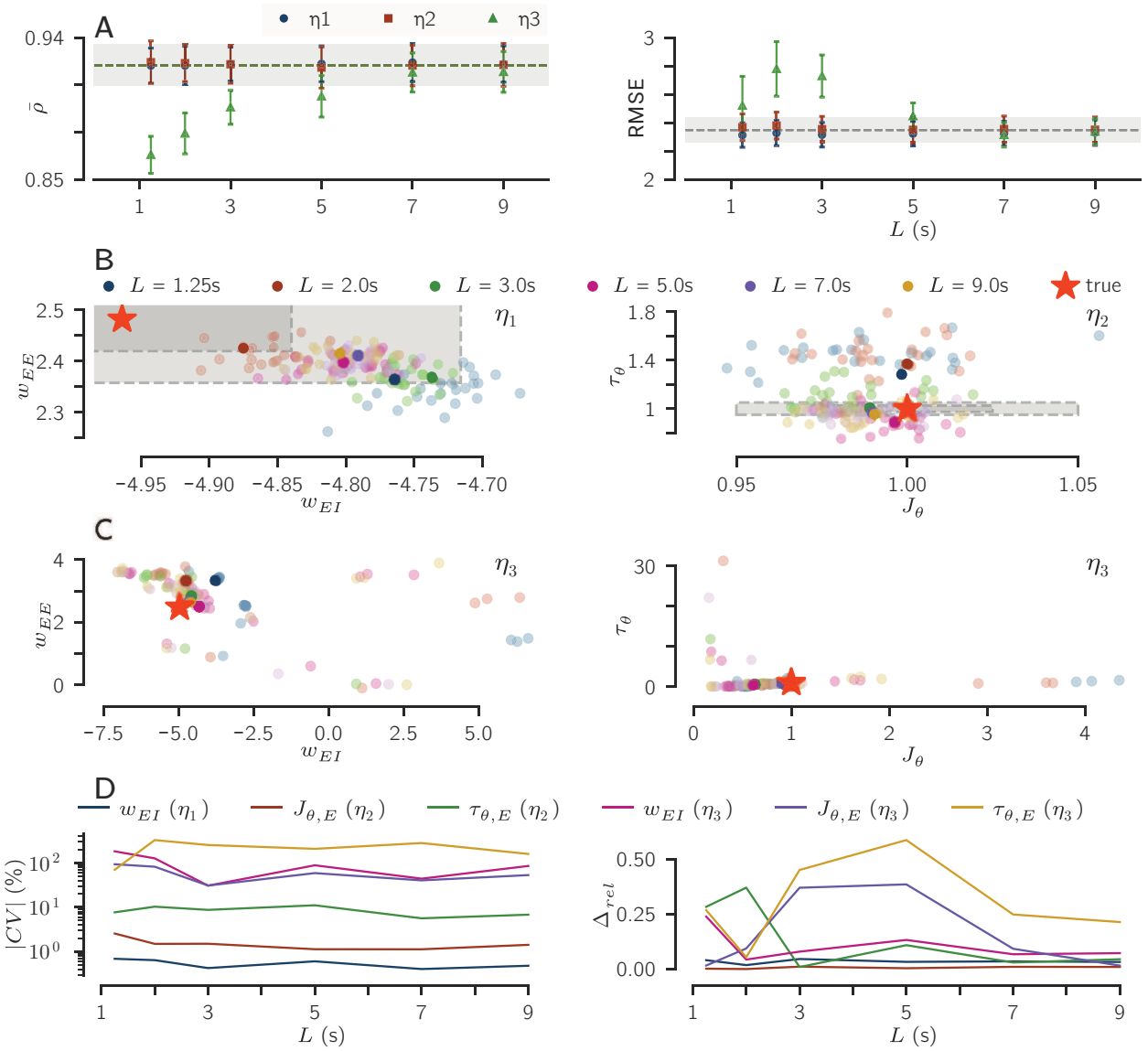


Figure 4.9.: Fits of many parameters are less consistent. a. As the number of inferred parameters is increased, more data is required to estimate them. (η_1, η_2 are parameter sets corresponding respectively to connectivity and adaptation. $\eta_3 \supset (\eta_1 \cup \eta_2)$ is the set of all parameters. Definitions in Table 4.11.) **b.** Results from 24 fits for subsets η_1 (left) and η_2 (right) for different amounts L of data. Star indicates the true parameters, and gray boxes the 5 and 10% relative errors Δ_{rel}. Fits cluster around the MAP, which for finite amounts of data will not exactly coincide with the ground truth values. Darker dots indicate the fit with the highest likelihood. The consistency of estimates for the adaptation parameters, with $\tau_{\theta,E} = 1$ s, is particularly improved with longer data traces. **c.** Same as in (b) but all parameters were simultaneously inferred. The reduced consistency is noticeable by the change of scale, at which the 5% and 10% relative error boxes are not visible. **d.** When going from inferring smaller (η_1, η_2) to larger (η_3) subsets of parameters, the increase in relative error for the same number of fits is relatively modest (right) compared to the wider area in parameter space to which fits converge (left). Figure traces statistics for different parameters as a function of the amount of data (L) and the subset of parameters that were fit simultaneously (η_1–η_3). Values for all data/subset combinations are given in Tables 4.12 and 4.13.

4.E. Mesoscopic update equations

This appendix first describes the quantities composing the state vector for the mesoGIF model, then lists the equations used for this paper. All equations are for discretized time, and we use a superscript (k) to indicate the k-th time step. For derivations and a more complete discussion of the variables involved, see Schwalger, Deger, and Gerstner [20].

4.E.1. Construction of the state vector

In order to obtain a finite state-vector (see Section 4.4.2), neurons are divided into two categories: "free''and "refractory"; the assignment of neurons to either category changes over time, following a discretized form of the transport equation (4.18).

Refractory neurons are still in the absolute or relative refractory period caused by their last spike, and thus have a higher firing threshold. Since the height of the threshold is dependent on that spike's time, we track a vector $m_\alpha^{(k)}$, indexed by the age l. We define the scalar $m_{\alpha,l}^{(k)}$ as our estimate of the number of neurons at time t_k thatd at time t_{k-l}. A vector $v_\alpha^{(k)}$ similarly tracks the variance of that estimate. The adaptation of the neurons depends on their age, such that their firing rate is also given by a vector, $\Lambda_\alpha^{(k)}$. With an adaptation time scale τ_θ of 1 s and time steps of 1 ms, these vectors each comprise around $K = 1000$ age bins. For a more detailed discussion on properly choosing K, see Equation (86) in Schwalger, Deger, and Gerstner [20].

Free neurons, meanwhile, have essentially forgotten their last spike: their firing threshold has relaxed back to its resting state, and so they can be treated as identical, independent of when that last spike was. One scalar per population, $\Lambda_{\text{free},\alpha}^{(k)}$, suffices to describe their firing rate. Scalars $x_\alpha^{(k)}$ and $z_\alpha^{(k)}$ respectively track the estimated mean and variance of the number of free neurons.

In the case of an infinite number of neurons, the firing rates $\Lambda_\alpha^{(k)}$ and $\Lambda_{\text{free},\alpha}^{(k)}$ would be exact, but for finite populations a further correction $P_{\Lambda,\alpha}^{(k)}$ must be made to account for statistical fluctuations. Combining $\Lambda_{\text{free},\alpha}^{(k)}$, $\Lambda_\alpha^{(k)}$ and $P_{\Lambda,\alpha}^{(k)}$, one can compute $\bar{n}_\alpha^{(k)}$, the expected number of spikes at t_k. The definition of $n^{(k)}$ then follows as described in Section 4.4.2.

For both refractory and free neurons, the dependency of their time evolution on the spiking history of the network is taken into account by convolving the population activities (one per population) with synaptic, membrane and adaptation kernels. Following Schwalger, Deger, and Gerstner [20], we express these as exponential filters; this allows the associated convolutions to be respectively replaced by three additional dynamic variables y, h and g, making forward simulations more efficient. Replacing temporal filters by dynamic variables has the additional important benefit of making the dynamics Markovian when we consider them as updates on a state $S^{(k)}$, composed of the concatenation of the blue variables in Figure 4.10,

$$S^{(k)} := \left(n^{(k)}, y^{(k)}, g^{(k)}, h^{(k)}, \mathbf{u}^{(\mathbf{k})}, \mathbf{\Lambda}^{(\mathbf{k})}, \Lambda_{\text{free}}^{(k)}, \mathbf{m}^{(\mathbf{k})}, \mathbf{v}^{(\mathbf{k})}, x^{(k)}, z^{(k)}\right) . \tag{4.42}$$

For clarity, we have here typeset in bold the components of $S^{(k)}$ with both population and age dimensions.

4.E.2. Update equations

The equations below follow from Schwalger, Deger, and Gerstner [20] after setting the synaptic filter to an exponential: $\epsilon_{\alpha\beta}(s) = \theta(s - \Delta_{\alpha\beta})e^{-(s-\Delta)/\tau_s,\beta}/\tau_s(\beta)$. They depend on the inverse link function f, relating membrane potential to spiking probability, and a refractory/adaptation kernel θ. Throughout this work we used

$$f_\alpha(u') = c_\alpha \exp(u'/\Delta_{\text{u},\alpha}) \tag{4.43}$$

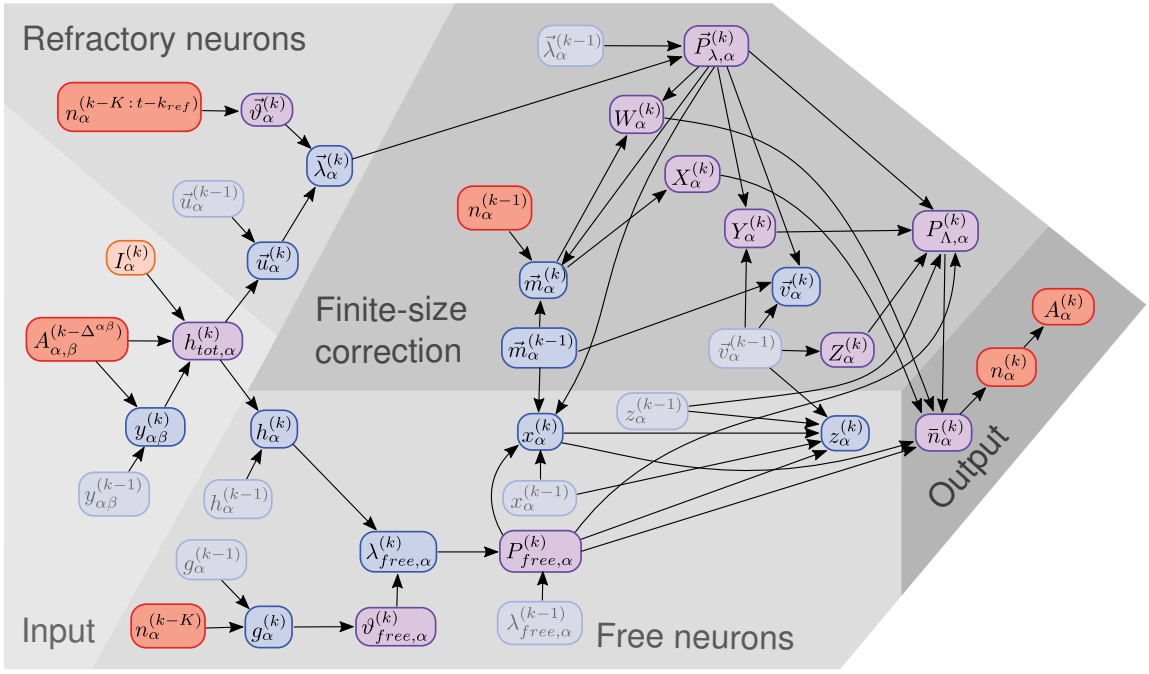


Figure 4.10.: Graphical representation of the mesoscopic model. An arrow $x \to y$ indicates that x is required in order to compute y. Red (orange) boxes indicate observed variables (input). Variables in blue boxes must be stored until the next iteration, and along with the activity A, form the model's state. Intermediate variables shown in purple do not need to be stored. Indices in parentheses indicate the time step, greek letters the population index. *During simulation*, mesoscopic model parameters (not shown, but determine the computations along arrows) are fixed, and mesoscopic output variables $A_\alpha^{(k)}$ are generated iteratively for each time step k; their values form the input for the next time step. *During inference*, the input is obtained from the training data, and is used to compute the sequence of binomial means $\bar{n}_\alpha^{(k)}$. These outputs, along with the observed outputs in the training data, are used to compute the likelihood. The gradient descent algorithm then changes the model parameters after each batch of training data to maximize the likelihood. See also Schwalger, Deger, and Gerstner [20, Figure 12].

and

$$\theta_\alpha(t) = \begin{cases} \infty & \text{if } t < t_{\text{ref},} \\ \frac{J_{\theta,\alpha}}{\tau_{\theta,\alpha}} e^{(t-t_{\text{ref},})/\tau_{\theta,\alpha}} & \text{otherwise.} \end{cases} \tag{4.44}$$

The quasi-renewal kernel [41] used below is defined as[†]

$$\tilde{\theta}_\alpha(t) = \Delta_{\text{u},\alpha} \left[1 - e^{-\theta_\alpha(t)/\Delta_{\text{u},\alpha}} \right] . \tag{4.45}$$

State vectors assign the index 0 to the time Δt, such that they run from $\theta_0 = \theta(\Delta t)$ to $\theta_K = \theta((K+1)\Delta t)$, with $K \in \mathbb{N}$. We define k_{ref} to be the lengths of the absolute refractory periods in time bins, i.e. $t_{\text{ref},} = k_{\text{ref},\alpha} \Delta t$ for each population α.

Total input

$$h^{(k+1)} = u_{rest} + (u^{(k)} - u_{rest}) e^{-\Delta t/\tau_m} + h_{tot} , \tag{4.46}$$

$$y_{\alpha\beta}^{(k+1)} = A_\beta(t_k - \Delta_{\alpha\beta}) + \left[y_{\alpha\beta}^{(k)} - A_\beta(t_k - \Delta_{\alpha\beta}) \right] e^{-\Delta t/\tau_{s,\beta}} . \tag{4.47}$$

[†] The double exponential obtained when we substitute Equation (4.44) into Equation (4.45) comes from the fact that the quasi-renewal kernel is also doubly integrated: at each time point, it is integrated over neuron ages.

$$h_{tot,\alpha} = RI_{ext}^{(k)}\left(1-e^{-\Delta t/\tau_{m,\alpha}}\right) + \tau_{m,\alpha}\sum_{\beta=1}^{M} p_{\alpha\beta}N_\beta w_{\alpha\beta}\Bigg\{A_\beta(t-\Delta_{\alpha\beta}) + \frac{\tau_{s,\beta}\,e^{-\frac{\Delta t}{\tau_{s,\beta}}}\left[y_{\alpha\beta}^{(k)} - A_\beta(t_k-\Delta_{\alpha\beta})\right] - e^{-\frac{\Delta t}{\tau_{m,\alpha}}}\left[\tau_{s,\beta}y_{\alpha\beta}^{(k)} - \tau_{m,\alpha}A_\beta(t_k-\Delta_{\alpha\beta})\right]}{\tau_{s,\beta}-\tau_{m,\alpha}}\Bigg\} \tag{4.48}$$

Membrane potential, refractory neurons

$$u_{\alpha,i}^{(k+1)} = \begin{cases} u_{r,\alpha} & 0 \le i < k_{\mathrm{ref},\alpha}, \\ u_{rest,\alpha} + (u_{\alpha,i-1}^{(k)} - u_{rest,\alpha})e^{-\Delta t/\tau_{m,\alpha}} + h_{tot,\alpha}^{(k)} & i \ge k_{\mathrm{ref},\alpha}. \end{cases} \tag{4.49}$$

Firing threshold

$$\vartheta_{\alpha i}^{(k+1)} = \vartheta_{\mathrm{free},\alpha}^{(k+1)} + \theta_{\alpha i} + \frac{1}{N}\sum_{j=i+1}^{K}\tilde{\theta}_{\alpha,j}\Delta n_\alpha^{(k-j-1)}, \tag{4.50}$$

$$\vartheta_{\mathrm{free},\alpha}^{(k+1)} = u_{th,\alpha} + J_{\theta,\alpha}e^{-T/\tau_{\theta,\alpha}}g^{(k+1)}, \tag{4.51}$$

$$g_\alpha^{(k+1)} = e^{-\Delta t/\tau_{\theta,\alpha}}g_\alpha^{(k)} + (1-e^{-\Delta t/\tau_{\theta,\alpha}})A_\alpha^{(k+1-K)}. \tag{4.52}$$

Firing probabilities

$$\Lambda_{\mathrm{free},\alpha}^{(k)} = f(h_\alpha^{(k)} - \vartheta_{\mathrm{free},\alpha}^{(k)}), \qquad \Lambda_{\alpha i}^{(k)} = \begin{cases} 0 & 0 \le i < k_{\mathrm{ref},\alpha}, \\ f(u_{\alpha i}^{(k)} - \vartheta_{\alpha i}^{(k)}) & k_{\mathrm{ref},\alpha} \le i < K. \end{cases} \tag{4.53}$$

$$P_{\mathrm{free},\alpha}^{(k)} = 1 - e^{-\bar{\Lambda}_{\mathrm{free},\alpha}^{(k)}\Delta t}, \qquad P_{\Lambda,\alpha i}^{(k)} = 1 - e^{-\bar{\Lambda}_{\alpha i}^{(k)}\Delta t}, \tag{4.54}$$

where

$$\bar{\Lambda}_{\mathrm{free},\alpha}^{(k)} = [\Lambda_{\mathrm{free},\alpha}^{(k-1)} + \Lambda_{\mathrm{free},\alpha}^{(k)}]/2, \tag{4.55}$$

$$\bar{\Lambda}_{\alpha i}^{(k)} = [\Lambda_{\alpha,i-1}^{(k-1)} + \Lambda_{\alpha i}^{(k)}]/2. \tag{4.56}$$

Survival counts

$$\bar{n}_\alpha^{(k)} = \sum_{i=0}^{K-1} P_{\Lambda,\alpha i}^{(k)}\bar{m}_{\alpha i}^{(k)} + P_{\mathrm{free},\alpha}^{(k)}x_\alpha^{(k)} + P_{\Lambda,\alpha}^{(k)}\left(N_\alpha - \sum_{i=0}^{K-1}\bar{m}_{\alpha i}^{(k)} - x_\alpha^{(k)}\right), \tag{4.57}$$

$$a_\alpha^{(k)} = \frac{\bar{n}_\alpha^{(k)}}{N_\alpha\Delta t}, \tag{4.58}$$

where

$$P_{\Lambda,\alpha}^{(k)} = \frac{\sum_{i=0}^{K-1}P_{\Lambda,\alpha i}^{(k)}v_{\alpha i}^{(k)} + P_{\mathrm{free}}^{(k)}z_\alpha^{(k)}}{\sum_{i=0}^{K-1}v_{\alpha i}^{(k)} + z_\alpha^{(k)}}, \tag{4.59}$$

$$x_\alpha^{(k)} = \sum_{i=K}^{\infty} \bar{m}_{\alpha i}^{(k)} = (1 - P_{\text{free},\alpha}^{(k)}) x_\alpha^{(k-1)} + m_{\alpha K}^{(k)}, \tag{4.60}$$

$$z_\alpha^{(k)} = \sum_{i=K}^{\infty} v_{\alpha i}^{(k)} = (1 - P_{\text{free},\alpha}^{(k)})^2 z_\alpha^{(k-1)} + P_{\text{free},\alpha}^{(k)} x_\alpha^{(k-1)} + v_{\alpha K}^{(k)}, \tag{4.61}$$

$$\bar{m}_{\alpha i}^{(k)} = \begin{cases} n_\alpha^{(k-1)} & \text{if } i = 0, \\ \left[1 - P_{\Lambda,\alpha i}^{(k)}\right] \bar{m}_{\alpha,i-1}^{(k-1)} & \text{otherwise;} \end{cases} \tag{4.62}$$

$$v_{\alpha i}^{(k)} = \begin{cases} 0 & \text{if } i = 0, \\ \left[1 - P_{\Lambda,\alpha i}^{(k)}\right]^2 v_{\alpha,i-1}^{(k-1)} + P_{\Lambda,\alpha i}^{(k)} \bar{m}_{\alpha i-1}^{(k-1)} & \text{otherwise.} \end{cases} \tag{4.63}$$

Spike generation

$$n_\alpha^{(k)} \sim \text{Binom}\left(\bar{n}_\alpha^{(k)} / N_\alpha; N_\alpha\right). \tag{4.64}$$

This last equation is the one identified as Equation (4.26) in the main text.

4.F. Performance of four population models

Table 4.14.: Performance of four population models – per-trial RMSE (Equation (4.33)). Measures computed from 60 realizations of each model.

			RMSE			
Input	Model		L2/3e	L2/3i	L4e	L4i
Sine	True	– micro	1.39 ± 0.03	3.46 ± 0.10	3.47 ± 0.13	4.59 ± 0.13
	Theory	– meso	1.39 ± 0.04	3.37 ± 0.10	3.76 ± 0.15	4.51 ± 0.13
	MAP	– meso	1.39 ± 0.03	3.49 ± 0.09	3.46 ± 0.13	4.53 ± 0.13
OU	True	– micro	1.22 ± 0.03	3.14 ± 0.08	2.26 ± 0.07	5.13 ± 0.15
	Theory	– meso	1.21 ± 0.03	3.06 ± 0.09	2.26 ± 0.07	4.95 ± 0.15
	MAP	– meso	1.22 ± 0.03	3.11 ± 0.08	2.25 ± 0.06	5.30 ± 0.14
Impulse	True	– micro	1.54 ± 0.05	3.64 ± 0.11	5.32 ± 0.46	5.11 ± 0.23
	Theory	– meso	1.59 ± 0.06	3.63 ± 0.11	7.99 ± 0.66	5.88 ± 0.36
	MAP	– meso	1.70 ± 0.07	3.96 ± 0.13	7.74 ± 0.59	6.00 ± 0.36

			$\bar{\rho}$			
Input	Model		L2/3e	L2/3i	L4e	L4i
Sine	True	– micro	0.418	0.386	0.994	0.948
	Theory	– meso	0.354	0.348	0.991	0.945
	MAP	– meso	0.352	0.455	0.994	0.951
OU	True	– micro	0.829	0.694	0.977	0.905
	Theory	– meso	0.815	0.717	0.978	0.914
	MAP	– meso	0.855	0.756	0.977	0.916
Impulse	True	– micro	0.914	0.879	0.996	0.927
	Theory	– meso	0.880	0.858	0.979	0.870
	MAP	– meso	0.912	0.896	0.988	0.887

Table 4.15.: Performance of four population models – trial-averaged correlation (Equation (4.34)). Measures computed from 60 realizations of each model.

4.G. Posterior for the 2 population mesoscopic model

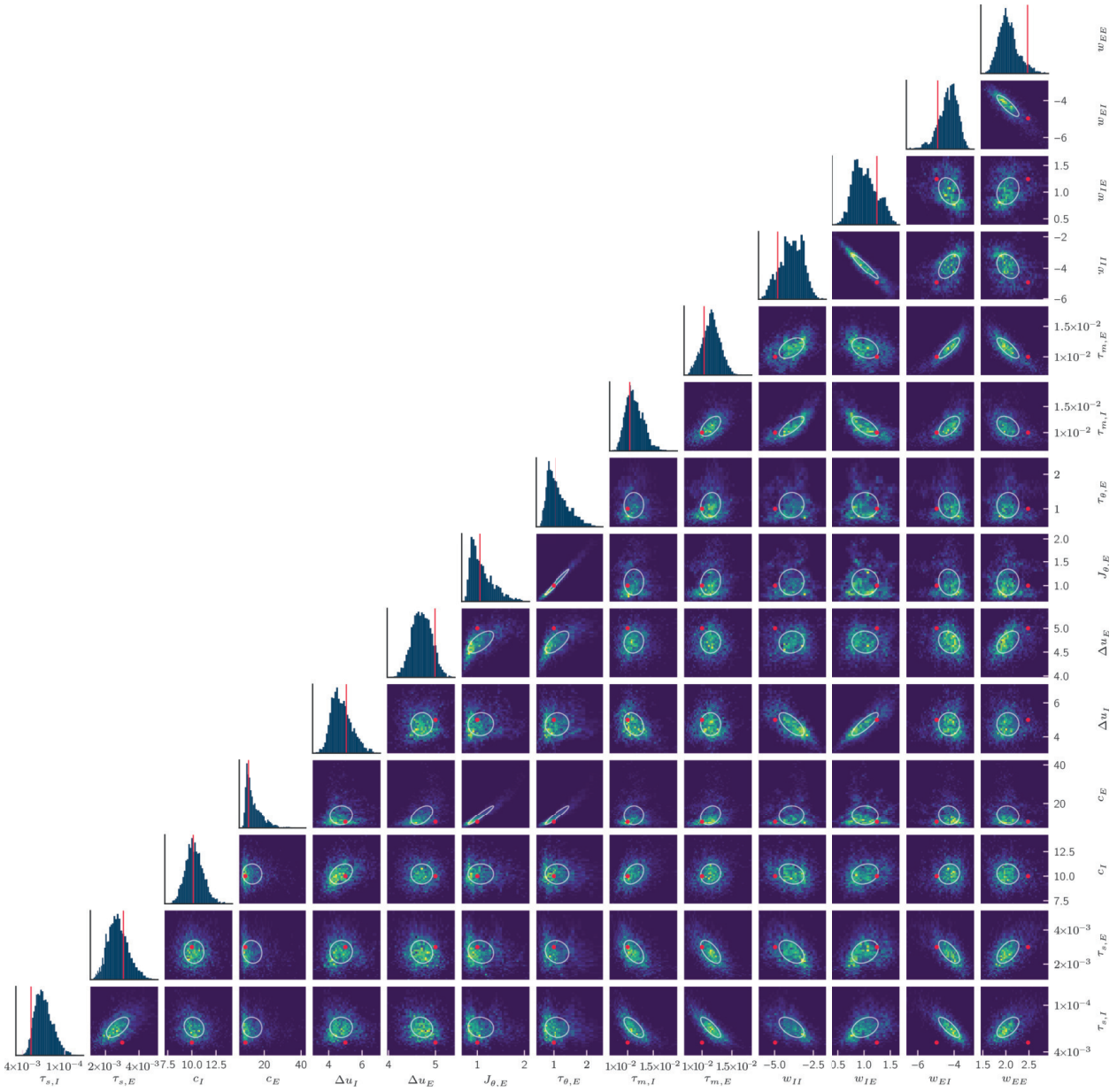


Figure 4.11.: Full posterior for the two population mesoscopic model. Red point indicates the true values and ellipses trace the two standard-deviation isoline assuming a Gaussian model. Many parameter pairs show noticeable correlation, such as $\tau_{\theta,E}$ and $J_{\theta,E}$, or w_{IE} and $\Delta_{\mathrm{u},I}$.

4.H. Fit dynamics

When fitting to data produced with a homogeneous microscopic model, inferred parameters are consistent with those predicted by the mesoscopic theory.

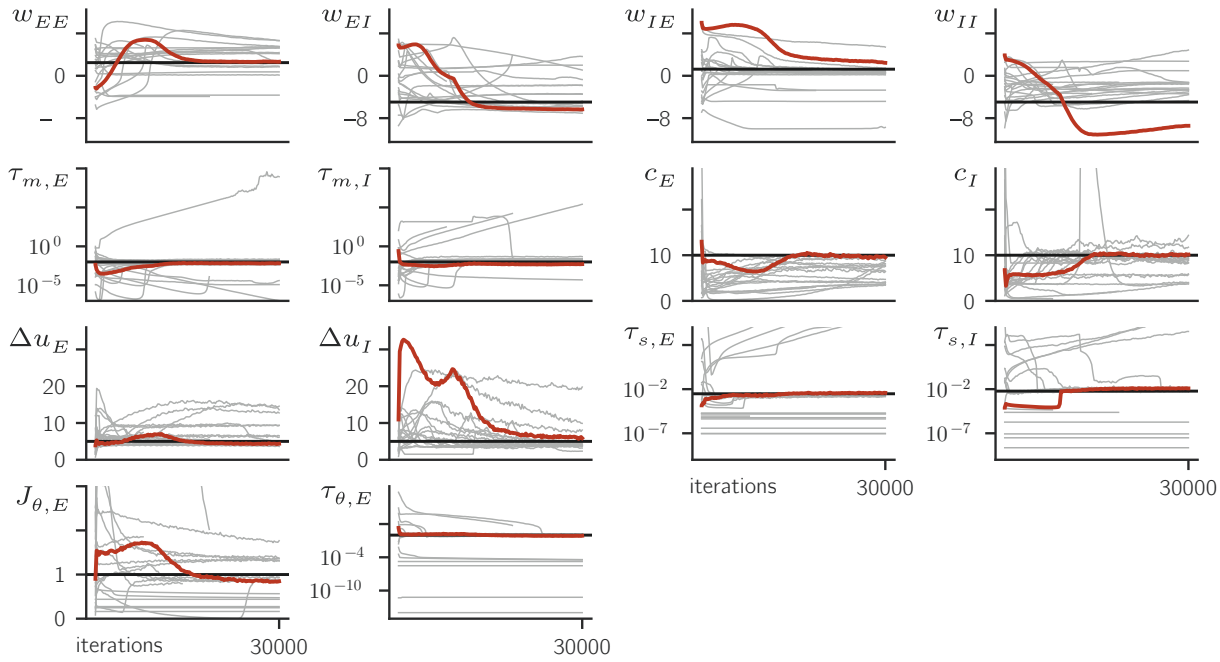


Figure 4.12.: Fit dynamics for the two population model. The 25 fits used to infer parameters for the two population model in Section 4.2.2. Fits in red are those that resulted in a likelihood within 5 orders of magnitude of the maximum, with brighter red indicating closer to the maximum. A total of 14 parameters were inferred; black lines indicate theoretical values.

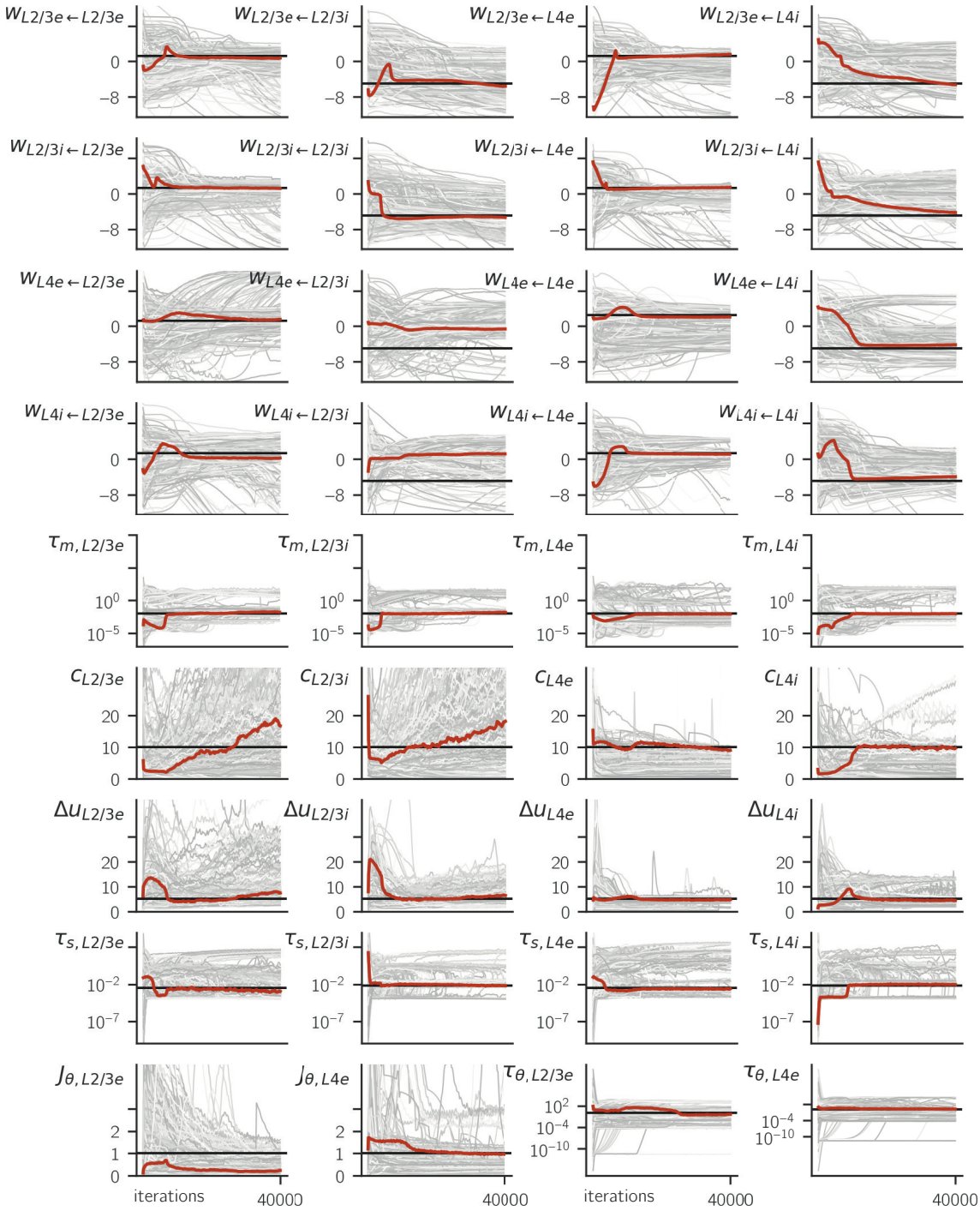


Figure 4.13.: Fit dynamics for the four population model. The 622 fits used to infer parameters for the four population model in Section 4.2.6. Although certain parameters would benefit from more iterations (e.g. c), most have converged within 4×10^4 iterations. A total of 36 parameters were inferred; black lines indicate theoretical values.

4.I. Self-consistent equation for the mesoscopic stationary state

We derive the stationary state for the case where $I_{ext} \equiv 0$. For analytical tractability, we assume that finite-size fluctuations are negligible (effectively, that N_α is very large), such that in the stationary state the activity is constant. We denote this activity A^*.

Having no fluctuations means that expected and observed spikes are interchangeable and equal to a constant;

$$n_\alpha^{(k)} = \bar{n}_\alpha^{(k)} = A^* N_\alpha \Delta t \,. \tag{4.65}$$

This means that the number of spikes never overshoots nor undershoots $\bar{n}_\alpha$, and the correction factor P_Λ is zero. Equivalently, we can state that

$$N = \sum_{i=0}^{K-1} \bar{m}_i + x \,. \tag{4.66}$$

Substituting the stationary values $A^*, h^*, \ldots$ into the equations of the previous appendix, we obtain equations for the stationary state. For instance,

$$h^* = u_{rest,\alpha} + \tau_m \sum_{\beta=1}^{M} p_{\alpha\beta} N_\beta w_{\alpha\beta} A^*_{\alpha\beta} \,, \tag{4.67}$$

$$y^*_{\alpha\beta} = A^*_{\alpha\beta} \,, \tag{4.68}$$

$$\vdots$$

and so on. Combining these with Equation (4.66), we obtain a self-consistency relation,

$$1 = A^*_\alpha \Delta t \left\{ k_{\mathrm{ref},\alpha} + 1 + \sum_{i=k_{\mathrm{ref},\alpha}}^{K-1} \exp\left[- \sum_{j=k_{\mathrm{ref},\alpha}+1}^{i-1} f(a_{\alpha j} + b^\beta_{\alpha j} A^*_\beta - c_{\alpha j} A^*_\alpha)\Delta t \right] + \frac{\exp\left[- \sum_{j=k_{\mathrm{ref},\alpha}}^{K-1} f(a_{\alpha j} + b^\beta_{\alpha j} A^*_\beta - c_{\alpha j} A^*_\alpha)\Delta t \right]}{1 - \exp\left[-f(a'_\alpha + b'^\beta_\alpha - c'_\alpha A^*_\alpha)\Delta t \right]} \right\} , \tag{4.69}$$

where $k_{\mathrm{ref},\alpha}$ is the number of bins corresponding to the absolute refractory period of population α. The terms therein are given by

$$a_{\alpha j} = e^{-(j - k_{\mathrm{ref},\alpha}+1)\Delta t/\tau_{m,\alpha}} (u_{r,\alpha} - u_{rest,\alpha}) + u_{rest,\alpha} - u_{th,\alpha} - \theta_{\alpha j} \,, \tag{4.70}$$

$$b^\beta_{\alpha j} = (1 - e^{-(j - k_{\mathrm{ref},\alpha}+1)\Delta t/\tau_{m,\alpha}}) \frac{1 - e^{-\Delta t/\tau_{m,\beta}}}{1 - e^{-\Delta t/\tau_{m,\alpha}}} \tau_m p^\beta_\alpha N^\beta w^\beta_\alpha \,, \tag{4.71}$$

$$c_{\alpha j} = J_{\theta,\alpha} e^{-T/\tau_{\theta,\alpha}} + \Delta t \sum_{j'=j+1}^{K} \tilde{\theta}_{\alpha j'} \,, \tag{4.72}$$

$$a'_\alpha = u_{rest,\alpha} - u_{th,\alpha} \,, \tag{4.73}$$

$$b'^\beta_\alpha = (1 - e^{-\Delta t/\tau_m}) \tau_m p^\beta_\alpha N^\beta w^\beta_\alpha \,, \tag{4.74}$$

$$c'_\alpha = J_{\theta,\alpha} e^{-T/\tau_{\theta,\alpha}} \,; \tag{4.75}$$

and the inverse link function f and the kernels θ and $\tilde{\theta}$ are as in Appendix 4.E.

Equation (4.69) can be solved numerically for A^*, after which the other state variables are easily calculated from the expressions in Appendix 4.E. We used SciPy's [130] `root` function with an initial guess of $A^*_\alpha = 1$ to solve for A^*. Since the stationary initialization was ultimately only used this to validate Algorithm 1 (see Appendix 4.C), we did no further analysis of Equation (4.69), and in particular leave the determination of conditions for which its solutions are unique to future work.

5. Epistemically robust selection of fitted models

Fitting models to data is an important part of the practice of science. Advances in machine learning have made it possible to fit more—and more complex—models, but have also exacerbated a problem: when multiple models fit the data equally well, which one(s) should we pick? The answer depends entirely on the modelling goal. In the scientific context, the essential goal is *replicability*: if a model works well to describe one experiment, it should continue to do so when that experiment is replicated tomorrow, or in another laboratory. The selection criterion must therefore be robust to the variations inherent to the replication process. In this work we develop a nonparametric method for estimating uncertainty on a model's empirical risk when replications are non-stationary, thus ensuring that a model is only rejected when another is *reproducibly* better. We illustrate the method with two examples: one a more classical setting, where the models are structurally distinct, and a machine learning-inspired setting, where they differ only in the value of their parameters. We show how, in this context of replicability or "epistemic uncertainty", it compares favourably to existing model selection criteria, and has more satisfactory behaviour with large experimental datasets.

Published as Alexandre René and André Longtin. "Selecting Fitted Models under Epistemic Uncertainty Using a Stochastic Process on Quantile Functions". In: *Nature Communications* 16.1 (Oct. 23, 2025)
https://doi.org/10.1038/s41467-025-64658-7
Online version https://alcrene.github.io/emd-paper

5.1. Introduction

Much of our understanding of the natural world is built upon mathematical models. But how we build those models evolves as new techniques and techonologies are developed. With the arrival of machine learning methods, it has become even more feasible to solve inverse problems and learn complex descriptions by applying data-driven methods to scientifically-motivated models [96, 139, 140]. However, even moderately complex models are generally *non-identifiable*: many different parameter sets may yield very similar outputs [21, 141, 142]. With imperfect real-world data, it is often unclear which—if any—of these models constitutes a trustable solution to the inverse problem.

This presents us with a model selection problem that is somewhat at odds with the usual statistical theory. First, we require a criterion that can distinguish between structurally identical models differing only in their *specific parameters*. Moreover, even with the most tightly controlled experiments, we expect all candidate models to be *misspecified* due to experimental variations. Those variations also *change* when the experiment is repeated in the same lab or another lab; in other words, the replication process is *non-stationary*, even when the data-generation process is stationary within individual trials. While there has been some work to develop model selection criteria that are robust to misspecification [74, 144, 145], and some consideration of locally stationary processes [146], the question of non-stationary replications has received little attention. Criteria that assume the data generating process to be stationary therefore underestimate *epistemic uncertainty*, which has important repercussions when they are used for scientific induction, where the goal is to learn a model that *generalises*. In recent years, a few authors have advanced similar arguments in the context of machine learning models [147, 148].

Terminology

In this chapter, we use the term **replication** to mean (definition from Barba [143]):
A study that arrives at the same scientific findings as another study, collecting new data (possibly with different methods) and completing new analyses.
This contrasts with the weaker form of a **reproduction** study, which repeats the analysis of another study on the *same* data to find the same results.
The concepts of **robustness** in statistics and **out-of-distribution generalisation** in machine learning are closely related to replicability, while **in-distribution generalisation** is closer to reproducibility.

Readers should be aware that unfortunately the terms **replication** and **reproduction** sometimes appear with opposite meanings in the literature [143].

To support data-driven scientific methods, there is therefore a need for a criterion that more accurately estimates this uncertainty due to differences between our model and the true data generating process. Epistemic uncertainty may be due to learning the model from limited samples, the model being misspecified or non-identifiable, or the data generating process being non-stationary. We distinguish this from *aleatoric* uncertainty (also called sampling uncertainty) [149, 150], which is due to the intrinsic stochasticity of the process.

Some of the pioneering work on this front came from the geosciences, which methods like GLUE [151–153] and Bayesian calibration [154]. More recently, we have also seen strategies to account for epistemic uncertainty in machine learning [150, 155], astrophysics [156] and condensed matter physics [157]. All of these employ some form of ensemble: epistemic uncertainty is represented as a concrete distribution over models, and predictions are averaged over this distribution.

Unfortunately, ensemble models are difficult to interpret since they are generally invalid in a Bayesian sense [79]. In our view, if the goal is to find interpretable models, then an approach like that advocated by Tarantola [142] seems more appropriate: instead of assigning probabilities to an ensemble of plausible models, simply treat that ensemble as the final inference result. With a finite (and hopefully small) number of plausible models, each can be interpreted individually.

The high-level methodology Tarantola proposes comes down to the following: 1. Construct a (potentionally large) set of *candidate models*. 2. Apply a *rejection criterion* to each candidate model in the set. 3. Retain from the set of candidate models those that satisfy the criterion.

Step 1 can be accomplished in different ways; for example, Prinz, Bucher, and Marder [21] performed a grid search using a structurally fixed model of the lobster's pyloric rhythm, and found thousands of distinct parameter combinations that reproduce chosen features of the recordings. More recently, machine learning methods have also been used to learn rich mechanistic models of molecular forces [139], cellular alignment [140] and neural circuits [24, 96]. Since these are intrinsically nonlinear models, their objective landscape contains a multitude of local minima, which translates to a multitude of candidate models. For a general formulation of the learning problem that fits our framing especially well, see Vapnik's *Function estimation model* [75].

The focus of our work is to present a practical criterion for step 2: **we therefore assume that we have already obtained a set of candidate models**. We make only three hard requirements. First, a candidate model $\mathcal{M}$ must be probabilistic, taking the form

$$p(y_i \mid x_i; \mathcal{M}), \quad (x_i, y_i) \in \mathcal{D}_{\text{test}}, \tag{5.1}$$

where $\mathcal{D}_{\text{test}}$ is the observed dataset and $(x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m)$ is the i-th input/output pair (or independent/dependent variables) that was observed. Often this takes the form of a mechanistic model with some additional observation noise. Second, it must be possible to generate arbitrarily many synthetic sample pairs (x, y) following each candidate model's distribution. And third, any candidate model $\mathcal{M}$ must assign a non-vanishing probability to each of the observations:

$$p(y_i \mid x_i; \mathcal{M}) > 0\,, \quad \forall (x_i, y_i) \in \mathcal{D}_{\text{test}}\,. \tag{5.2}$$

For example, a model whose predictions y are restricted to an interval $[a, b]$ is only allowed if all observations are within that interval. We also require the definition of a loss function Q to quantify the accuracy of model predictions y_i. Taking the expectation of Q over data samples yields the *risk* R, a standard measure of performance used to fit machine learning models.

Key to our approach is a novel method for assigning to each model a *risk-* or *R-distribution* representing epistemic uncertainties due either to finite samples, model misspecification or non-stationarity. Only when two models have sufficiently non-overlapping R-distributions do we reject the one with higher R. This approach allows us to compare any number of candidate models, by reducing the problem to a sequence of transitive pairwise comparisons. We make no assumption on the structure of those models: they may be given by two completely different sets of equations, or they may have the same equations but differ only in their parameters, as long as they can be cast in the form of equation (5.1).

In many cases, models will contain a "physical" component—the process we want to describe—and an "observation" component—the unavoidable experimental noise. Distinguishing these components is often useful, but it makes no difference from the point of view of our method: only the combined "physical + observation" model matters. In their simplest forms, the physical component may be deterministic and the observation component may be additive noise, so that the model is written as a deterministic function f plus a random variable ξ affecting the observation:

$$y_i = f(x_i; \mathcal{M}) + \xi_i\,.$$

In such a case, the probability in equation (5.1) reduces to $p(\xi_i = y_i - f(x_i; \mathcal{M}))$; if ξ_i is Gaussian with variance σ^2, this further reduces to $p(y_i \mid x_i; \mathcal{M}) \propto \exp\!\left(-(y_i - f(x_i; \mathcal{M}))^2 / 2\sigma^2\right)$. Of course, in many cases the model itself is stochastic, or the noise is neither additive nor Gaussian. Moreover, even when such assumptions seem justified, we should be able to test them against alternatives.

We will illustrate our proposed criterion using two examples from biology and physics. The first compares candidate models of neural circuits (from the aforementioned work of Prinz, Bucher, and Marder [21]) with identical structure but different parameters; the second compares two structually different models of black body radiation, inspired by the well-known historical episode of the "ultraviolet catastrophe".

In the next few sections, we present the main steps of our analysis using the neural circuit model for illustration. We start by arguing that modelling discrepancies are indicative of epistemic uncertainty. We then use this idea to assign uncertainty to each model's risk, and thereby define the *EMD rejection rule*. Here EMD stands for *empirical modelling discrepancy*, which describes the manner in which we estimate epistemic uncertainty; this mainly involves three steps, schematically illustrated in Fig. 5.1. First, we represent the prediction accuracy of each model with a quantile function q^* of the loss. Second, by measuring the self-consistency of q^* with the model's own predictions $(\tilde{q})$, we obtain a measure (δ^{EMD}) of epistemic uncertainty, from which we construct a stochastic process $\mathfrak{Q}$ over quantile functions. Since each realisation of $\mathfrak{Q}$ can be integrated to yield the risk, $\mathfrak{Q}$ thus induces the R-distributions we seek. This requires however the introduction of a new type of stochastic process, which we call *hierarchical beta process*, in order to ensure that realizations are valid quantile functions. Those R-distributions are used to compute the tail probabilities, denoted B^{EMD}, in terms of which the rejection rule is defined. The third step is a calibration procedure, where we validate the estimated probabilities B^{EMD} on a set of simulated experiments. To ensure the soundness of our results, we used over 24,000 simulated experiments, across 16 forms of experimental variation.

The following sections then proceed with a systemic study of the method, using the simpler model of black body radiation. We illustrate how modelling errors and observation noise interact to affect the shape of R distributions and ultimately the EMD rejection criterion. Finally, we compare our method with a number of other popular criteria—AIC, BIC, MDL, elpd and Bayes factors. We highlight the major modelling assumptions each method makes, and therefore

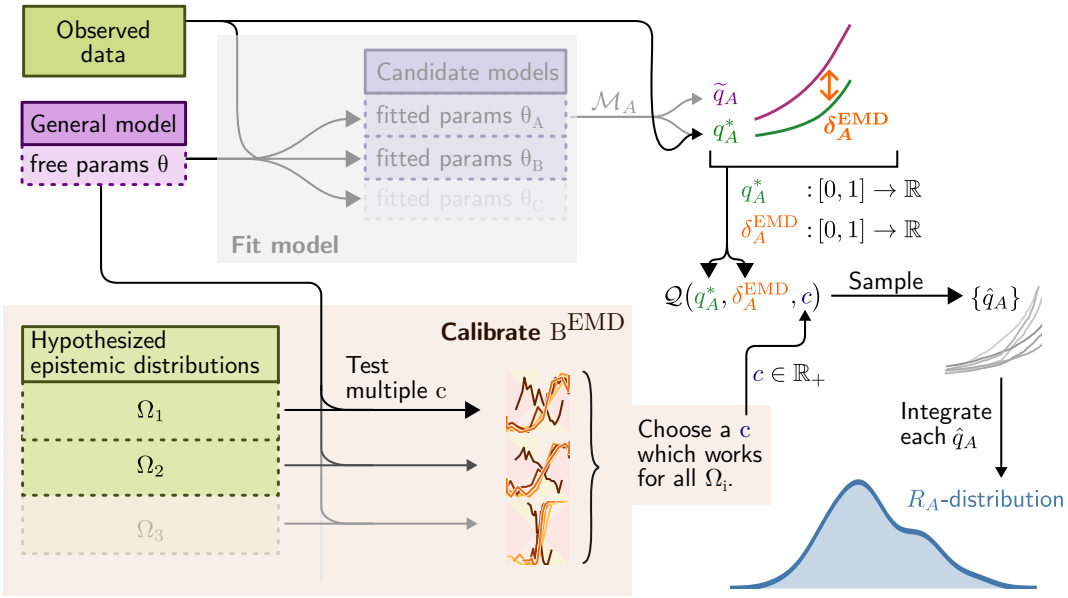


Figure 5.1.: Overview of the approach for computing R-distributions. We assume the model has already been fitted to obtain a set of *candidate parameter sets* θ_A, θ_B...(Not shown: the models may also be structurally different.) Each candidate parameter set Θ_A defines a *candidate model* $\mathcal{M}_A$, for which we describe the statistics of the loss with two different quantile functions: the purely synthetic $\tilde{q}_A$ (Equation (5.22)) which depends only on the model, and the mixed q_A^* (Equation (5.19)) which depends on both the model and the data. A small discrepancy δ_A^{EMD} (equation (5.23)) between those two curves indicates that model predictions concord with the observed data. Both q_A^* and δ_A^{EMD} are then used to parametrise a stochastic process $\mathfrak{Q}$ that generates random quantile functions. This induces a distribution for the risk R_A of model $\mathcal{M}_A$, which we ascribe to *epistemic uncertainty*. The stochastic process $\mathfrak{Q}$ also depends on a global scaling parameter c. This is independent of the specific model, and is obtained by calibrating the procedure with simulated experiments Ω_i that reflect variations in laboratory conditions. The computation steps on the right (white background) have been packaged as available software [158].

in which circumstances it is likely most appropriate. Special attention is paid to the behaviour of the model selection statistics as the sample size grows, which is especially relevant to the scientific context.

5.2. Results

5.2.1. The risk as a selection criterion for already fitted models

As set out in the Introduction, we start from the assumption that models of the natural phenomenon of interest have already been fitted to data; this anterior step is indicated by the grey faded rectangle in Fig. 5.1. The manner in which this is done is not important; model parameters for example could result from maximizing the likelihood, running a genetic algorithm or performing Bayesian inference.

Our starting points are the true data-generating process $\mathcal{M}_{\mathrm{true}}$ and the pointwise loss Q. The loss depends on the model $\mathcal{M}_A$, evaluates on individual data samples (x_i, y_i) and returns a real number:

$$Q(x_i, y_i; \mathcal{M}_A) \in \mathbb{R}\,. \tag{5.3}$$

Often, but not always, the negative log likelihood is chosen as the loss. What we seek to estimate is the risk, i.e. the expectation of Q:

$$R_A = \mathbb{E}_{(x_i,y_i)\sim\mathcal{M}_{\mathrm{true}}}[Q(x_i, y_i; \mathcal{M}_A)]; \tag{5.4}$$

we use the notation $(x_i, y_i) \sim \mathcal{M}_{\mathrm{true}}$ to indicate that the samples (x_i, y_i) are drawn from $\mathcal{M}_{\mathrm{true}}$. For simplicity, our notation assumes that model parameters are point estimates. For Bayesian models, one can adapt the definition of the loss Q to include an expectation over the posterior; equation (5.4) would then become the Bayes risk. Our

methodology is agnostic to the precise definition of Q, as long as it is evaluated pointwise; it can differ from the objective used to fit the models.

The risk describes the expected performance of a model on replicates of the dataset, when each replicate is drawn from the same data-generating process $\mathcal{M}_{\text{true}}$. It is therefore a natural basis for comparing models based on their ability to generalise. It is also the gold standard objective for a machine learning algorithm [48, 75], for the same reason, and has been used in classical statistics to analyse model selection under misspecification [88]. In practice we usually estimate R_A from finite samples with the *empirical risk*:

$$\hat{R}_A = \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} Q(x_i, y_i; \mathcal{M}_A)\,. \tag{5.5}$$

For the types of problems we are interested in, it is safe to assume that $\hat{R}_A$ converges to R_A in probability.

A common consideration with model selection criteria is whether they are *consistent*, i.e. whether they eventually select the true model when the number of samples grows. When models are misspecified, none of the candidates are actually the true model, so the definition of consistency is often generalised to mean convergence to the "pseudo-true" model: the one with the smallest Kullback-Leibler divergence D_{KL} from the true model [93, 159]. If consistency in this sense is desired, then using the log likelihood for Q is recommended, since for a fixed true model, minimizing the D_{KL} is equivalent to minimizing the log likelihood. The log likelihood is also known to be *proper* (see e.g. references [50], [54] or [134] (Chap. 7.1)); a selection rule that minimizes the log likelihood is thus also consistent in the original sense of asymptotically selecting the true model when it is among the candidates.

The statistical learning literature instead defines a "learning machine" as *consistent* if it asymptotically minimizes the true risk R when trained with the empirical risk $\hat{R}_A$ [48, 75]. As long as we have a finite number of candidate models, our procedure will satisfy this notion of consistency.

Convergence to R_A also means that the result of a risk-based criterion is insensitive to the size of the dataset, provided we have enough data to estimate the risk of each model accurately. This is especially useful in the scientific context, where we want to select models that can be used on datasets of any size.

In addition to defining a consistent, size-insensitive criterion, risk also has the important practical advantage that it remains informative when the models are structurally identical. We contrast this with the marginal likelihood (also known as the model evidence), which for some dataset $\mathcal{D}$, model $\mathcal{M}_A$ parametrised by θ, and prior π_A, would be written

$$\mathcal{E}_A = \int p\big(\, \mathcal{D} \,\big|\, \mathcal{M}_A(\theta) \,\big)\, \pi_A(\theta) d\theta\,. \tag{5.6}$$

Since it is integrated over the entire parameter space, $\mathcal{E}_A$ characterizes the family of models $\mathcal{M}_A(\cdot)$, but says nothing of a specific parametrisation $\mathcal{M}_A(\theta)$. We include a comparison of our proposed EMD rejection rule to other common selection criteria, including the model evidence, at the end of our Results. .

A criterion based on risk however will not account for a model which overfits the data, which is why it is **important to use separate datasets for fitting and comparing models**. This in any case better reflects the usual scientific paradigm of recording data, forming a hypothesis, and then recording new data (either in the same laboratory or a different one) to test that hypothesis. Splitting data into training and test sets is also the standard practice in machine learning to avoid overfitting and ensure better generalisation. In the following therefore we will always denote the observation data as $\mathcal{D}_{\text{test}}$ to stress that they should be independent from the data to which the models were fitted.

In short, the EMD rejection rule we develop in the following sections ranks models based on their empirical risk equation (5.5), augmented with an uncertainty represented as a risk-distribution. Comparisons based on the risk are consistent and insensitive to the number of test samples, but require a separate test dataset to avoid overfitting.

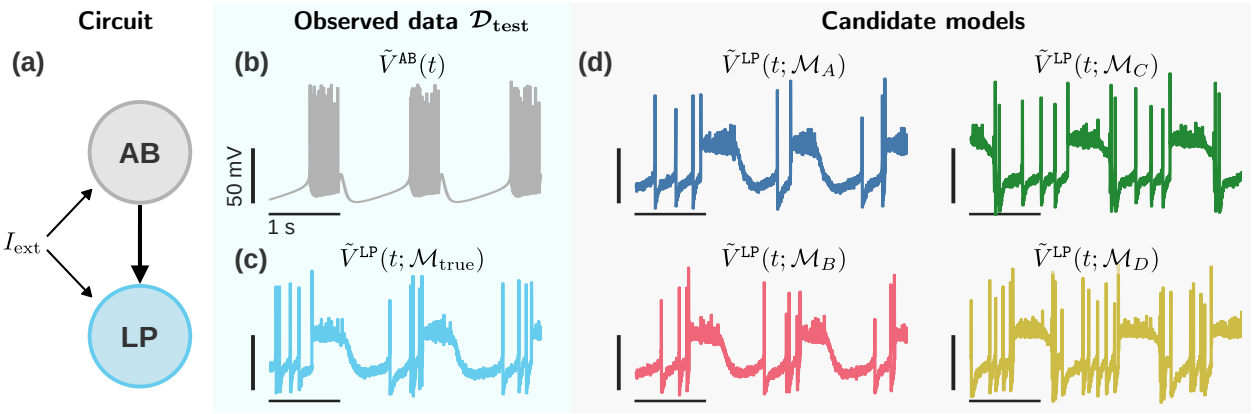


Figure 5.2.: Comparison of LP neuron responses The goal is to estimate a good set of candidate models for neuron LP. **a)** We consider a simple circuit with a known pacemaker neuron (AB) and a post-synaptic neuron of unknown response (LP). **b)** Output of the AB neuron. This serves as input to the LP neuron. In practice, these data would more likely be provided by an experimental recording of neuron AB, but here we assume it is a simulatable known model for convenience. **c)** Response of neuron model `LP 1` to the input in (b). Together, (b) and (c) serve as our observed data $\mathcal{D}_{\text{test}}$. **d)** Response of neuron models `LP 2` to `LP 5` to the input in (b). These are our four candidate models of neuron LP.

5.2.2. Example application: Selecting among disparate parameter sets of a biophysical model

To explain the construction of the EMD rejection rule, we use the dynamical model for a neuron membrane potential described by Prinz, Bucher, and Marder [21]. This choice was motivated by the fact that fitting a neuron model is a highly ill-posed problem, and therefore corresponds to the situation we set out in the Introduction: disparate sets of parameters for a structurally fixed model that nevertheless produce similar outputs. We will focus on the particular LP neuron type; Prinz, Bucher, and Marder [21] find five distinct parameter sets that reproduce its experimental characteristics. Throughout this work, we reserve `LP 1` as the model that generates through simulation the *true data* $\mathcal{D}_{\text{test}}$, and use `LP 2` to `LP 5` to define the *candidate models* $\mathcal{M}_A$ to $\mathcal{M}_D$ that we compare against $\mathcal{D}_{\text{test}}$. We intentionally exclude `LP 1` from the candidate parameters to emulate the typical situation where none of the candidate models fit the data perfectly. Visual inspection of model outputs suggests that two candidates (models $\mathcal{M}_A$ and $\mathcal{M}_B$) are more similar to the true model output (Fig. 5.2). We will show that our method not only concords with those observations, but makes the similarity between models quantitative.

It is important to note that we treat the models $\mathcal{M}_A$ to $\mathcal{M}_D$ as simply given. We make no assumption as to how they were obtained, or whether they correspond to the maximum of a likelihood. We chose this neural example, where parameters are obtained with a multi-stage, semi-automated grid search, partly to illustrate that our method is agnostic to the fitting procedure.

The possibility of comparing models visually was another factor in choosing this example. Since numbers can be misleading, this allows us to confirm that the method works as expected. Especially for our target application where all candidate models share the same equation structure, a visual validation is key to establishing the soundness of the B^{EMD}, since none of the established model selection like the Bayes factor, AIC or WAIC [56, 134] are applicable. Indeed, these other methods only compare alternative equation structures, not alternative parameter sets. We include a more in-depth comparison to these other methods at the end of the Results.

The datasets in this example take the form of one-dimensional time series, with the time $t \in \mathcal{T}$ as the independent variable and the membrane potential $V^{\text{LP}} \in \mathcal{V}$ as the dependent variable. We denote the space of all possible time-potential tuples $\mathcal{T}\times\mathcal{V}$. Model specifics are given in the Methods; from the point of view of model selection, what matters is that we have ways of generating series of these time-potential tuples: either using the true data-generating process ($\mathcal{M}_{\text{true}}$) or one of the candidate models ($\mathcal{M}_A$ to $\mathcal{M}_D$).

We will assume that the dataset used to evaluate models is composed of L samples, with each sample a $(t, \tilde{V}^{\text{LP}})$ tuple:

$$\mathcal{D}_{\text{test}} = \left\{\left(t_k, \tilde{V}^{\text{LP}}(t_k; \mathcal{M}_{\text{true}}, I_{\text{ext}})\right) \in \mathcal{T}\times\mathcal{V}\right\}_{k=1}^{L}. \tag{5.7}$$

The original model by Prinz, Bucher, and Marder [21] produced deterministic traces V^{LP}. Experimental measurements however are variable, and our approach depends on that variability. For this work we therefore augment the model with two sources of stochasticity. First, the system as a whole receives a coloured noise input I_{ext}, representing an external current received from other neurons. (External currents may also be produced by the experimenter, to help model the underlying dynamics.) Second, we think of the system as the combination of a biophysical model—described by Equations (5.50) to (5.52)—and an observation model that adds Gaussian noise. The observation model represents components that don't affect the biophysics—like noise in the recording equipment—and can be modelled to a first approximation as:

$$\tilde{V}^{\mathrm{LP}}(t_k; \mathcal{M}_{\mathrm{true}}) = \underbrace{V^{\mathrm{LP}}\big(t_k, I_{\mathrm{ext}}(t_k; \tau, \sigma_i); \mathcal{M}_{\mathrm{true}}\big)}_{\text{biophysical model}} + \underbrace{\xi(t_k; \sigma_o)}_{\text{observation model}} . \tag{5.8}$$

The parameters of the external input I_{ext} are τ and σ_i, which respectively determine its autocorrelation time and strength (i.e. its amplitude). The observation model has only one parameter, σ_o, which determines the strength of the noise. The symbol $\mathcal{M}_{\mathrm{true}}$ is shorthand for everything defining the data-generating process, which includes the parameters τ, σ_i and σ_o. The indices i and o are used to distinguish *input* and *output* model parameters.

Since the candidate models in this example assume additive observational Gaussian noise, a natural choice for the loss function is the negative log likelihood:

$$\begin{aligned} Q(t_k, \tilde{V}^{\mathrm{LP}}; \mathcal{M}_a) &= -\log p(\tilde{V}^{\mathrm{LP}} \mid t_k, \mathcal{M}_a) \\ &= \frac{1}{2}\log(2\pi\sigma_o) - \frac{\big(\xi_k\big)^2}{2\sigma} , \\ &= \frac{1}{2}\log(2\pi\sigma_o) - \frac{\big(\tilde{V}^{\mathrm{LP}} - V^{\mathrm{LP}}(t_k; \mathcal{M}_a)\big)^2}{2\sigma_o} . \end{aligned} \tag{5.9}$$

In Equation (5.9), $a \in \{A, B, C, D\}$ is the candidate model index and ξ_k is the observation noise at time t_k.

However one should keep in mind that a) it is not necessary to define the loss in terms of the log likelihood, and b) the loss used to *compare* models need not be the same used to *fit* the models. For example, if the log likelihood of the assumed observation model is non-convex or non-differentiable, one might use a simpler objective for optimization. Alternatively, one might fit using the log likelihood, but compare models based on global metrics like the interspike interval. Notably, Prinz, Bucher, and Marder [21] do not directly fit to potential traces $\tilde{V}$, but rather use a set of data-specific heuristics and global statistics to select candidate models. Gonçalves et al. [24] similarly find that working with specially crafted summary statistics—rather than the likelihoods—is more practical for inferring this type of model. In this work we nevertheless stick with the generic form of equation (5.9), which makes no assumptions on the type of data used to fit the models and is thus easier to generalise to different scenarios.

The definition of the risk then follows immediately:

$$R_a \coloneqq \mathbb{E}_{(t,\tilde{V}^{\mathrm{LP}})\sim\mathcal{M}_{\mathrm{true}}}\big[Q(t, \tilde{V}^{\mathrm{LP}}; \mathcal{M}_a)\big] . \tag{5.10}$$

Here a stands for one of the model labels A, B, C or D. The notation $(t, \tilde{V}^{\mathrm{LP}}) \sim \mathcal{M}_{\mathrm{true}}$ denotes that the distribution from which the samples $(t, \tilde{V}^{\mathrm{LP}})$ are drawn is $\mathcal{M}_{\mathrm{true}}$. We can think of $\mathcal{M}_{\mathrm{true}}$ as producing an infinite* sequence of data points by integrating the model dynamics and appyling equation (5.8) multiple times, or by going to the laboratory and performing the experiment multiple times.

As alluded to above, the candidate model traces in Fig. 5.2**d** suggest two groups of models: the $\mathcal{M}_A$ and $\mathcal{M}_B$ models seem to better reproduce the data than $\mathcal{M}_C$ or $\mathcal{M}_D$. Within each group however, it is hard to say whether one model is better than the other; in terms of the risks, this means that we expect $R_A, R_B < R_C, R_D$. This also means that we should be wary of a selection criterion that unequivocally ranks $\mathcal{M}_A$ better than $\mathcal{M}_B$, or $\mathcal{M}_C$ better than $\mathcal{M}_D$.

In other words, we expect that uncertainties on R_A and R_B should be at least commensurate with the difference $|R_A - R_B|$, and similarly for the uncertainties on R_C and R_D.

* The sequence of data points produced by a real-world experiment of course cannot be truly infinite, but it is unbounded since we can almost always run the experiment for longer, or repeat it more times.

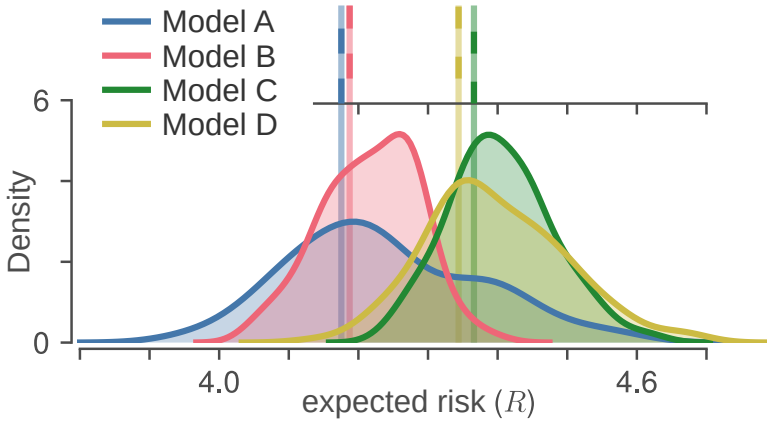


Figure 5.3.: Empirical risk vs. R-distributions for the four candidate LP models. (top) The empirical risk (5.5) for each of the four candidate LP models. **(bottom) Our proposed B^{EMD} criterion** replaces the risk by an *R-distribution*, where the spread of each distribution is due to the *replication uncertainty* for that particular model. R-distributions are distributions of the R functional in equation (5.20); we estimate them by sampling the quantile function (i.e. inverse cumulative density function) q according to a stochastic process $\mathfrak{Q}$ on quantile functions. We used an EMD sensitivity factor of $c = 2^{-2}$ (see later section on calibration) for $\mathfrak{Q}$ and drew samples $\hat{q} \sim \mathfrak{Q}$ until the relative standard error on the risk was below 3 %. A kernel density estimate (KDE) is used to display those samples as distributions. The R-distribution for the true model is much narrower (approximately Dirac) and far to the left, outside the plotting bounds.

5.2.3. A conceptual model for replicate variations

Evaluating the risk (5.10) for each candidate model on a dataset $\mathcal{D}_{\text{test}}$ yields four scalars $\hat{R}_A$ to $\hat{R}_D$; since a lower risk should indicate a better model, a simple naive model selection rule would be

$$\begin{cases} \text{reject } \mathcal{M}_b & \text{if } \hat{R}_a < \hat{R}_b\,, \\ \text{reject } \mathcal{M}_a & \text{if } \hat{R}_a > \hat{R}_b\,, \\ \text{no rejection} & \text{if } \hat{R}_a = \hat{R}_b\,, \end{cases} \qquad \text{for } a, b \in \{A, B, C, D\}\,. \tag{5.11}$$

As with many selection criteria (e.g. AIC [160, 161], BIC [86], DIC [87, 162]), this rule is effectively binary: the third option, to reject neither model, has probability zero. It therefore *always* selects either $\mathcal{M}_a$ or $\mathcal{M}_b$, even when the evidence favouring one of the two is extremely weak. Another way to see this is illustrated at the top of Fig. 5.3: the lines representing the four values R_A through R_D have no error bars, so even minute differences suffice to rank the models.

The problem is that from a scientific standpoint, if the evidence is too weak, this ranking is likely irrelevant—or worse, misinformative. Hence our desire to assign uncertainty to each estimate of the risk. Ideally we would like to be able to compute tail probabilities like $P(R_A < R_B)$, which would quantify the strength of the evidence in favour of either $\mathcal{M}_A$ or $\mathcal{M}_B$. Selecting a minimum evidence threshold ϵ would then allow us to convert equation (5.11) into a true ternary decision rule with non-zero probability of keeping both models:

$$\begin{cases} \text{reject } \mathcal{M}_b & \text{if } P(R_a < R_b) > \epsilon\,, \\ \text{reject } \mathcal{M}_a & \text{if } P(R_a < R_b) < (1 - \epsilon)\,, \\ \text{no rejection} & \text{if } (1 - \epsilon) \leq P(R_a < R_b) \leq \epsilon\,. \end{cases} \tag{5.12}$$

As we explain in the Introduction, our goal is to select a model that can describe not just the observed data $\mathcal{D}_{\text{test}}$, but also new data generated in a replication experiment. In order for our criterion to be *robust*, the tail probabilities $P(R_a < R_b)$ should account for uncertainty in the replication process.

Note that for a given candidate model $\mathcal{M}_a$ and fixed $\mathcal{M}_{\text{true}}$, we can always estimate the risk with high accuracy if we have enough samples, irrespective of the amount of noise intrinsic to $\mathcal{M}_a$ or of misspecification between $\mathcal{M}_a$ and $\mathcal{M}_{\text{true}}$. Crucially however, we *also do not assume the replication process to be stationary*, so a replicate dataset $\mathcal{D}'_{\text{test}}$ may be drawn from a slightly different data-generating process $\mathcal{M}'_{\text{true}}$. This can occur even when $\mathcal{M}_{\text{true}}$ itself

is stationary, reflecting the fact that it is often easier to control variability within a single experiment than across multiple ones. Ontologically therefore, the uncertainty on $\hat{R}$ is a form of *epistemic uncertainty* arising from the variability across (experimental) replications.

To make this idea concrete, consider that the input I_{ext} and noise ξ might not be stationary over the course of an experiment with multiple trials. We can represent this by making their parameters random variables, for example

$$\Omega \coloneqq \begin{cases} \log \sigma_o \sim \mathcal{N}(0.0\,\text{mV}, (0.5\,\text{mV})^2) \\ \log \sigma_i \sim \mathcal{N}(-15.0\,\text{mV}, (0.5\,\text{mV})^2) \\ \log_{10} \tau \sim \text{Unif}([0.1\,\text{ms}, 0.2\,\text{ms}]) \end{cases}, \tag{5.13}$$

and drawing new values of $(\sigma_o, \sigma_i, \tau)$ for each trial (i.e. each replicate). Since it is a distribution over data-generating processes, we call Ω an *epistemic distribution*. For illustration purposes, here we have parametrised Ω in terms of two parameters of the biophysical model and one parameter of the observation model, thus capturing epistemic uncertainty within a single experiment. In general the parametrisation of Ω is a modelling choice, and may represent other forms of non-stationarity—for example due to variations between experimental setups in different laboratories.

Conceptually, we could estimate the tail probabilities $P(R_a < R_b)$ by sampling J different data-generating processes $\mathcal{M}^j_{\text{true}}$ from Ω, for each then drawing a dataset $\mathcal{D}^j_{\text{test}} \sim \mathcal{M}^j_{\text{true}}$, and finally computing the empirical risks $\hat{R}^j_a$ and $\hat{R}^j_b$ on $\mathcal{D}^j_{\text{test}}$. The fraction of datasets for which $\hat{R}^j_a < \hat{R}^j_b$ would then estimate the tail probability:

$$P(R_a < R_b) \approx \frac{1}{J} \left| \left\{ j \;\middle|\; \hat{R}^j_a < \hat{R}^j_b \right\}_{j=1}^{J} \right|. \tag{5.14}$$

The issue of course is that we cannot know Ω. First because we only observe data from a single $\mathcal{M}_{\text{true}}$, but also because there may be different contexts to which we want to generalise: one researcher may be interested in modelling an individual LP neuron, while another might seek a more general model that can describe all neurons of this type. These two situations would require different epistemic distributions, with the latter one being in some sense broader.

However *we need not commit to a single epistemic distribution*: if we have two distributions, and we want to ensure that conclusions hold under both, we can instead use the condition

$$\min_{\Omega \in \{\Omega_1, \Omega_2\}} P_\Omega(R_a < R_b) > \epsilon \tag{5.15}$$

to attempt to reject $\mathcal{M}_b$. In general, considering more epistemic distributions will make the model selection more robust, at the cost of discriminatory power. *Epistemic distributions are not therefore prior distributions*, since a Bayesian calculation is always tied to a specific choice of prior. (The opposite however holds: a prior can be viewed as a particular choice of epistemic distribution.)

We view epistemic distributions mostly as conceptual tools. For calculations, we will instead propose in the next sections a different type of distribution ($\mathfrak{Q}$; technically a stochastic process), which is not on the data-generating process, but on the distribution of pointwise losses. Being lower-dimensional and more stereotyped than Ω, we will be able to construct $\mathfrak{Q}$ entirely non-parametrically, up to a scaling constant c (c.f. Fig. 5.1 and the section listing desiderata for $\mathfrak{Q}$). A later section we will then show, through numerical calibration and validation experiments, that $\mathfrak{Q}$ also has nice universal properties, so that the only thing that really matters is the overall scale of the epistemic distributions. The constant c is matched to this scale by numerically simulating epistemic distributions as part of the calibration experiments.

5.2.4. Model discrepancy as a baseline for non-stationary replications

To keep the notation in the following sections more general, we use the generic x and y as independent and dependent variables. To recover expressions for our neuron example, substitute $x \to t$, $y \to \tilde{V}^{\text{LP}}$, $\mathcal{X} \to \mathcal{T}$ and $\mathcal{Y} \to \mathcal{V}$. Where

possible we also use A and B as a generic placeholder for a model label.

Our goal is to define a selection criterion that is robust against variations between experimental replications, but which can be computed *using knowledge only of the candidate models and the observed empirical data*. To do this, we make the following assumption:

EMD assumption (version 1). *Candidate models represent that part of the experiment that we understand and control across replications.*

More precisely, in the next section we define the *empirical model discrepancy* function $\delta_A^{\mathrm{EMD}} : (0,1) \to \mathbb{R}_{\geq 0}$ such that if model $\mathcal{M}_A$ exactly reproduces the observations, then δ_A^{EMD} is identically zero. This function therefore measures the discrepancy between model predictions and actual observations. Since we expect misspecified models to have positive discrepancy ($\int_0^1 \delta_A^{\mathrm{EMD}}(\Phi) d\Phi > 0$), in the following we treat the discrepancy δ_A^{EMD} as a measure of *misspecification*.

Under our EMD assumption, misspecification in a model corresponds to experimental conditions we don't fully control, and which could therefore vary across replications of the experiment. Concretely this means that we can reformulate the EMD assumption as

EMD assumption (version 2). *The variability of R_A across replications is predicted by the model discrepancy δ_A^{EMD}.*

We also assume that the data-generating process $\mathcal{M}_{\mathrm{true}}$ within one replication is *strictly stationary* with finite correlations, i.e. that all observations are identically distributed but may be correlated. For simplicity in fact we treat the samples as i.i.d., since if necessary this could be done by thinning. See references [74] or [163] for discussions on constructing estimators from correlated time series.

Below we further assume a particular *linear* relationship between δ_A^{EMD} and the replication uncertainty: the function δ_A^{EMD} is scaled by the aforementioned sensitivity factor c to determine the variance of the stochastic process $\mathfrak{Q}_A$ (which we recall induces the R_A-distribution for the risk of model $\mathcal{M}_A$). The parameter $c \in \mathbb{R}_+$ therefore represents a conversion from *model discrepancy* to *epistemic uncertainty*. A practitioner can use this parameter to adjust the sensitivity of the criterion to misspecification: a larger value of c will emulate more important experimental variations. Since the tail probabilities (equation (5.14)) we want to compute will depend on c, we will write them

$$B_{AB;c}^{\mathrm{EMD}} \coloneqq P(R_A < R_B \mid c)\,. \tag{5.16}$$

The EMD rejection rule. *For a chosen rejection threshold $\epsilon \in (0.5, 1]$, reject model $\mathcal{M}_A$ if there exists a model $\mathcal{M}_B$ such that $B_{AB;c}^{\mathrm{EMD}} < \epsilon$ and $\hat{R}_A > \hat{R}_B$.*

The second condition ($\hat{R}_A > \hat{R}_B$) ensures the rejection rule remains consistent, even if the R-distributions become skewed. (See comment below equation (5.5).)

As an illustration, Table 5.1 gives the value of B^{EMD} for each candidate model pair in our example from Figures 5.2 and 5.3. As expected, models that were visually assessed to be similar also have B^{EMD} values close to $\frac{1}{2}$. In practice one would not necessarily need to compute the entire table, since the B^{EMD} satisfy *dice transitivity* [164, 165]. In particular this implies that for any threshold $\epsilon > \varphi^{-2}$ (where φ is the golden ratio), we have

$$\left.\begin{aligned} B_{AB;c}^{\mathrm{EMD}} &> \sqrt{\epsilon} \\ B_{BC;c}^{\mathrm{EMD}} &> \sqrt{\epsilon} \end{aligned}\right\} \Rightarrow\ B_{AC;c}^{\mathrm{EMD}} > \epsilon\,. \tag{5.17}$$

Since any reasonable choice of threshold will have $\epsilon > 0.5 > \varphi^{-2}$, we can say that whenever the comparisons $B_{AB;c}^{\mathrm{EMD}}$ and $B_{BC;c}^{\mathrm{EMD}}$ are both greater than $\sqrt{\epsilon}$, we can treat them as transitive. We give a more general form of this result in the Supplementary Methods.

	A	B	C	D
A	0.500	0.483	0.846	0.821
B	0.517	0.500	0.972	0.940
C	0.154	0.028	0.500	0.463
D	0.179	0.060	0.537	0.500

Table 5.1.: Comparison of B^{EMD} probabilities for candidate LP models. Values are the probabilities given by equation (5.16) with the candidate labels a and b corresponding to rows and columns respectively. Candidate models being compared are those of Fig. 5.2. Probabilities are computed for the R-distributions shown in Fig. 5.3, which used an EMD constant of $c = 2^{-2}$. Since $P(R_a < R_b) = 1 - P(R_b < R_a)$, the sum of symmetric entries equals 1.

5.2.5. δ^{EMD}: Expressing misspecification as a discrepancy between CDFs

We can treat the loss function for a given model $\mathcal{M}_A$ as a random variable $Q(x, y; \mathcal{M}_A)$ where the (x, y) are sampled from $\mathcal{M}_{\text{true}}$. A key realisation for our approach is that *the CDF (cumulative distribution function) of the loss suffices to compute the risk*. Indeed, we have for the CDF of the loss

$$\begin{aligned}\Phi_A^*(q) &:= p\big(Q(x, y; \mathcal{M}_A) \leq q \mid x, y \sim \mathcal{M}_{\text{true}}\big) \\ &= \int_{\mathcal{X}\times\mathcal{Y}} H\big(q - Q(x, y; \mathcal{M}_A)\big)\, p\big(x, y \mid \mathcal{M}_{\text{true}}\big)\, dxdy \\ &\approx \frac{1}{L} \sum_{x_i, y_i \in \mathcal{D}_{\text{test}}} H\big(q - Q(x_i, y_i; \mathcal{M}_A)\big)\,,\end{aligned} \tag{5.18}$$

where H is the Heaviside function with $H(q) = 1$ if $q \geq 0$ and 0 otherwise.

Since $\mathcal{M}_{\text{true}}$ is unknown, a crucial feature of 5.18 is that $\Phi_A^*(q)$ can be estimated without needing to evaluate $p\big(x, y \mid \mathcal{M}_{\text{true}}\big)$; indeed, all that is required is to count the number of observed data points in $\mathcal{D}_{\text{test}}$ that according to $\mathcal{M}_A$ have a loss less than q. Moreover, since Q returns a scalar, the data points (x_i, y_i) can have any number of dimensions: the number of data points required to get a good estimate of $\Phi_A^*(q)$ does not depend on the dimensionality of the data, for the same reason that estimating marginals of a distribution requires much fewer samples than estimating the full distribution. Finally, because the loss is evaluated using $\mathcal{M}_A$, but the expectation is taken with respect to a distribution determined by $\mathcal{M}_{\text{true}}$, we call Φ_A^* the *mixed CDF*.

We can invert Φ_A^* to obtain the *mixed PPF* (*percent point function*, also known as *quantile function*, or *percentile function*):

$$q_A^*(\Phi) := {\Phi_A^*}^{-1}(\Phi)\,, \tag{5.19, \textit{mixed PPF}}$$

which is also a 1-d function, irrespective of the dimensionality of $\mathcal{X}$ or $\mathcal{Y}$. We can then rewrite the risk as a one dimensional integral in q_A^*:

$$\begin{aligned}R_A = R[q_A^*] &= \int_0^1 q_A^*(\Phi)\, d\Phi \\ &\approx \frac{1}{L} \sum_{x_i, y_i \in \mathcal{D}_{\text{test}}} Q(x_i, y_i; \mathcal{M}_A)\,.\end{aligned} \tag{5.20}$$

To obtain equation (5.20), we simply used Fubini's theorem to reorder the integral of 5.18 and marginalized over all slices of a given loss $q_A^*(\Phi)$. The integral form (first line of equation (5.20)) is equivalent to averaging an infinite number of samples, and therefore to the *(true) risk*, whereas the average over observed samples (second line of equation (5.20)) is exactly the *empirical risk* defined in equation (5.5).

In practice, to evaluate equation (5.20), we use the observed samples to compute the sequence of per-sample losses $\{Q(x_i, y_i; \mathcal{M}_A)\}_{x_i, y_i \in \mathcal{D}_{\text{test}}}$. This provides us with a sequence of losses, which we use as ordinate values. We then sort this sequence so that we have $\{Q_i\}_{i=1}^L$ with $Q_i \leq Q_i + 1$, and assign to each the abscissa $\Phi_i = i/L + 1$, such that losses are motonically increasing and uniformly distributed on the $[0, 1]$ interval. This yields the *empirical PPF of the loss*—the "empirical" qualifier referring to this construction via samples, as opposed to an analytic calculation. Interpolating the points then yields a continuous function that can be used in further calculations. All examples in this paper linearly interpolate the PPF from $2^{10} = 1024$ points.

In Fig. 5.4 we show four examples of empirical PPFs, along with their associated empirical CDFs. We see that the statistics of the additive observational noise affects the shape of the PPF: for noise with exponential tails, as we

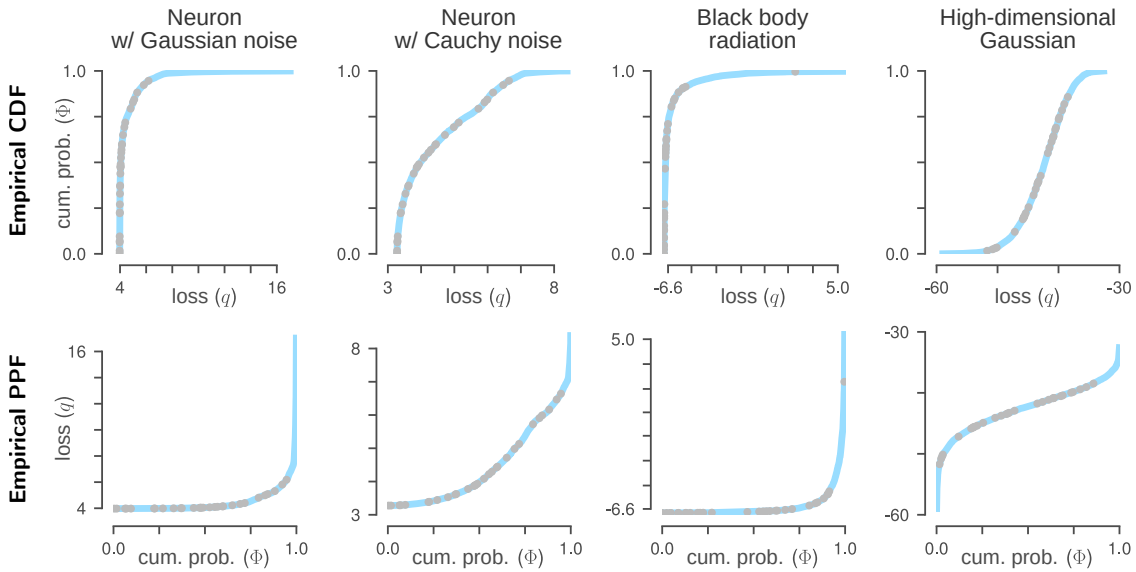


Figure 5.4.: Loss PPF for different models: each column corresponds to a different model. The PPF (bottom row) is the inverse of the CDF (top row). For calculations we interpolate $2^{10} = 1024$ points (cyan line) to obtain a smooth function; for illustration purposes here only 30 points are shown. The data for the first two columns were generated with the neuron model described at the top of our Results, where the additive noise follows either a Gaussian or Cauchy distribution. The black body radiation data for the third column were generated from a Poisson distribution using equation (5.38) with $s = 2^{14}$ and λ in the range 6 µm to 20 µm. Here the true noise is binomial, but the loss assumes a Gaussian. The fourth column shows an example where the data are high-dimensional; the same 30 dimensional, unit variance, isotropic Gaussian is used for both generating the data and evaluating the loss. In all panels the loss function used is the log likelihood under the model.

get from Gaussian or Poisson distributions, we have strong concentration around the minimum value of the loss followed by a sharp increase at $\Phi = 1$. For heavier-tailed distributions like Cauchy, loss values are less concentrated and the PPF assigns non-negligible probability mass to a wider range of values. The dimensionality of the data also matters. High-dimensional Gaussians are known to place most of their probability mass in a thin shell centered on the mode, and we see this in the fourth column of Fig. 5.4: the sharp increase at $\Phi = 0$ indicates that very low probability is assigned to the minimum loss.

Since by construction, the abscissae Φ of an empirical PPF are spaced at intervals of $1/L$, the Riemann sum for the integral in equation (5.20) reduces to the sample average. More importantly, we can interpret the risk as a functional in $q^*_A(\Phi)$, which will allow us below to define a generic stochastic process that accounts for epistemic uncertainty.

Up to this point with equation (5.20) we have simply rewritten the usual definition of the risk. Recall now that in the previous section, we proposed to equate replication uncertainty with misspecification; specifically we are interested in how differences between the candidate model $\mathcal{M}_A$ and the true data-generating process $\mathcal{M}_{\text{true}}$ affect the loss PPF, since this determines the risk. Therefore we also compute the PPF of $Q(x, y; \mathcal{M}_A)$ under its own model (recall from equation (5.1) that $\mathcal{M}_A$ must be a probabilistic model):

$$\begin{aligned}\widetilde{\Phi}_A(q) &\coloneqq p\big(Q(x, y; \mathcal{M}_A) \leq q \mid x, y \sim \mathcal{M}_A\big) \\ &= \int_{\mathcal{X}\times\mathcal{Y}} dxdy\; H\big(q - Q(x, y; \mathcal{M}_A)\big)\, p(x, y \mid \mathcal{M}_A) \\ &\approx \frac{1}{L_{\text{synth},A}} \sum_{x_i, y_i \in \mathcal{D}_{\text{synth},A}} H\big(q - Q(x_i, y_i; \mathcal{M}_A)\big)\,,\end{aligned} \tag{5.21}$$

from which we obtain the PPF:

$$\tilde{q}_A(\Phi) \coloneqq \widetilde{\Phi}_A^{-1}(\Phi)\,. \tag{5.22, \textit{synth PPF}}$$

The only difference between $\tilde{q}_A$ and q^*_A is the use of $p(x, y \mid \mathcal{M}_A)$ instead of $p(x, y \mid \mathcal{M}_{\text{true}})$ in the integral. In practice this integral would also be evaluated by sampling, using $\mathcal{M}_A$ to generate a dataset $\mathcal{D}_{\text{synth},A}$ with $L_{\text{synth},A}$ samples.

Because in this case the candidate model is used for both generating samples and defining the loss, we call $\tilde{q}_A$ ($\widetilde{\Phi}_A$) the *synthetic PPF* (*CDF*).

The idea is that the closer $\mathcal{M}_A$ is to $\mathcal{M}_{\text{true}}$, the closer also the synthetic PPF should be to the mixed PPF—indeed, equality of the PPFs ($\tilde{q}_a = q^*_A$) is a necessary condition for equality of the models ($\mathcal{M}_A = \mathcal{M}_{\text{true}}$). Therefore we can quantify the uncertainty due to misspecification, at least insofar as it affects the risk, as the absolute difference between $\tilde{q}_A$ and q^*_A:

$$\begin{aligned} \delta_A^{\text{EMD}} \;:\; [0,1] &\to \mathbb{R} \\ \Phi &\mapsto \left| \tilde{q}_A(\Phi) - q^*_A(\Phi) \right|. \end{aligned} \tag{5.23}$$

We refer to δ_A^{EMD} as the *empirical model discrepancy* (EMD) function because it measures the discrepancy between two empirical PPFs.

It is worth noting that even a highly stochastic data-generating process $\mathcal{M}_{\text{true}}$, with a lot of aleatoric uncertainty, still has a well defined PPF—which would be matched by an equally stochastic candidate model $\mathcal{M}_A$. Therefore the discrepancies measured by δ_A^{EMD} are a representation of the *epistemic* uncertainty. These discrepancies can arise either from a mismatch between $\mathcal{M}_A$ and $\mathcal{M}_{\text{true}}$, or simply having too few samples to estimate the mixed PPF q^*_A exactly; either mechanism contributes to the uncertainty on the expected risk of replicate datasets. We illustrate some differences between aleatoric and epistemic uncertainty in Supplementary Fig. 5.12, and include further comments on how different types of uncertainties relate to risk in the Supplementary Discussion.

5.2.6. $\mathfrak{Q}$: A stochastic process on quantile functions

When replication is non-stationary, each replicate dataset will produce a different loss PPF. A distribution over replications (like the previously defined epistemic distribution Ω) therefore corresponds to a distribution over PPFs. This leads us to the final formulation of our EMD assumption, which we now operationalize by assuming a linear relation between empirical model discrepancy and epistemic uncertainty:

EMD principle. *For a given model $\mathcal{M}_A$, differences between its PPFs on two replicate datasets should be proportional to δ_A^{EMD}.*

Formalising this idea is made considerably simpler by the fact that PPFs are always scalar functions, irrespective of the model or dataset's dimensionality. We do this as follows, going through the steps schematically represented by downward facing arrows on the right of Fig. 5.1:

> For a given model $\mathcal{M}_A$, we treat the PPFs of different replicates as realisations of a stochastic process $\mathfrak{Q}_A$ on the interval $[0,1]$; a realisation of $\mathfrak{Q}_A$ (i.e. a PPF) is a function $\hat{q}_A(\Phi)$ with $\Phi \in [0,1]$. The stochastic process $\mathfrak{Q}_A$ is defined to satisfy the desiderata listed below, which ensure that it is centered on the observed PPF q^*_A and that its variance at each Φ is governed by δ_A^{EMD}.

Obtaining the R-distributions shown in Fig. 5.3 is then relatively straightforward: we simply sample an ensemble of $\hat{q}_A$ from $\mathfrak{Q}_A$ and integrate each to obtain an ensemble of risks.

Working with PPFs has the important advantage that we can express our fundamental assumption as a simple linear proportionality relation between empirical model discrepancy and epistemic uncertainty. While this is certainly an approximation, it simplifies the interpretability of the resulting R-distribution. As we show in the Supplementary Discussion, the resulting criterion is also more robust than other alternatives. The simplicity of the assumption however belies the complexity of defining a stochastic process $\mathfrak{Q}$ directly on PPFs.

5.2.6.1. Desiderata for $\mathfrak{Q}$

In order to interpret them as such, realisations of $\hat{q}_A \sim \mathfrak{Q}_A$ must be valid PPFs. This places quite strong constraints on those realisations, for example:

- All realisations $\hat{q}_A(\Phi) \sim \mathfrak{Q}_A$ must be *monotone*.
- All realisations $\hat{q}_A(\Phi) \sim \mathfrak{Q}_A$ must be *integrable*.

Monotonicity follows immediately from definitions: a CDF is always monotone because it is the integral of a positive function equation (5.18), and therefore its inverse must also be monotone.

Integrability simply means that the integral in equation (5.20) exists and is finite. Concretely this is enforced by ensuring that the process $\mathfrak{Q}_A$ is *self-consistent* [166], a property which we explain in the Methods.

Interpreting the realisations $\hat{q}$ as PPFs also imposes a third constraint, more subtle but equally important:

- The process $\mathfrak{Q}_A$ must be *non-accumulating*.

A process that is accumulating would start at one end of the domain, say $\Phi = 0$, and sequentially accumulate increments until it reaches the other end. Brownian motion over the interval $[0, T]$ is an example of such a process. In contrast, consider the process of constructing a PPF for the data in Fig. 5.2: initially we have few data points and the PPF of their loss is very coarse. As the number of points increases, the PPF gets refined, but since loss values occur in no particular order, this happens *simultaneously across the entire interval*.

The accumulation of increments strongly influences the statistics of a process; most notably, the variance is usually larger further along the domain. This would not make sense for a PPF: if $\mathbb{V}[\mathfrak{Q}_A(\Phi)]$ is smaller than $\mathbb{V}[\mathfrak{Q}_A(\Phi')]$, that should be a consequence of $\delta^{\mathrm{EMD}}(\Phi)$ being smaller than $\delta^{\mathrm{EMD}}(\Phi')$—not of Φ occurring "before" Φ'.

This idea that a realisation $\hat{q}_A(\Phi)$ is generated simultaneously across the interval led us to define $\mathfrak{Q}_A$ as a sequence of *refinements*: starting from an initial increment $\Delta\hat{q}_1(0) = \hat{q}_A(1) - \hat{q}_A(0)$ for the entire Φ interval $[0, 1]$, we partition $[0, 1]$ into n subintervals, and sample a set of n subincrements in such a way that they sum to $\Delta\hat{q}_1(0)$. This type of distribution, where n random variables are drawn under the constraint of a fixed sum, is called a *compositional* distribution [167]. Note that the constraint reduces the number of dimensions by one, so a pair of increments would be drawn from a 1-d compositional distribution. A typical 1-d example is the beta distribution for $x_1 \in [0, 1]$, with $x_2 = (1 - x_1)$ and $\alpha, \beta > 0$:

$$\text{if } \; x_1 \sim \mathrm{B}(\alpha, \beta)\,, \qquad \text{then } \; p(x_1) \propto x_1^{\alpha-1}(1 - x_1)^{\beta-1}\,. \tag{5.24}$$

Interestingly, the most natural statistics for compositional distributions are not the mean and variance, but analogue notions of *centre* and *metric variance* [167, 168]; for the beta distribution defined above, these are

$$\mathbb{E}_a[(x_1, x_2)] = \frac{1}{e^{\psi(\alpha)} + e^{\psi(\beta)}}\big(e^{\psi(\alpha)}, e^{\psi(\beta)}\big) \tag{5.25a}$$

$$\mathrm{Mvar}[(x_1, x_2)] = \frac{1}{2}\big(\psi_1(\alpha) + \psi_1(\beta)\big)\,, \tag{5.25b}$$

where ψ and ψ_1 are the digamma and trigamma functions respectively, and $\mathbb{E}_a$ denotes expectation with respect to the Aitchison measure [167, 169]. In essence, Equations (5.25a) and (5.25b) are obtained by mapping x_1 and x_2 to the unbounded domain $\mathbb{R}$ via a logistic transformation, then evaluating moments of the unbounded variables. Of particular relevance is that—in contrast to the variance—the metric variance Mvar of a compositional distribution is therefore unbounded, which simplifies the selection of α and β (see Choosing beta distribution parameters in the Methods).

Of course, we not only want the $\hat{q}_A$ to be valid PPFs, but also descriptive of the model and data. We express this as two additional constraints, which together define a notion of closeness to q^*_A:

- At each intermediate point $\Phi \in (0, 1)$, the *centre* of the process is given by $q^*_A(\Phi)$:

$$\mathbb{E}_a\big[\hat{q}(\Phi)\big] = q^*_A(\Phi)\,. \tag{5.26}$$

- At each intermediate point $\Phi \in (0,1)$, the *metric variance* is proportional to the square of $\delta_A^{\mathrm{EMD}}(\Phi)$:

$$\mathrm{Mvar}\big[\hat{q}(\Phi)\big] = c\,\delta_A^{\mathrm{EMD}}(\Phi)^2\,. \tag{5.27}$$

Note that 5.27 formalises the EMD principle stated at the top of this section, with the sensitivity factor $c > 0$ determining the conversion from *empirical model discrepancy* (which we interpret as misspecification) to *epistemic uncertainty*. Note also that Equations (5.26) and (5.27) are akin to a variational approximation of the stochastic process around q_A^*. Correspondingly, we should expect these equations (and therefore $\mathfrak{Q}$) to work best when c is not too large. We describe how one might determine an appropriate value for c in the later section on calibration.

In addition to the above desiderata, for reasons of convenience, we also ask that

- The end points are sampled from Gaussian distributions:

$$\begin{aligned}\hat{q}(0) &\sim \mathcal{N}(q_A^*(0), c\,\delta_A^{\mathrm{EMD}}(0)^2)\,,\\ \hat{q}(1) &\sim \mathcal{N}(q_A^*(1), c\,\delta_A^{\mathrm{EMD}}(1)^2)\,.\end{aligned} \tag{5.28}$$

Thus the process $\mathfrak{Q}_{A;c}$ is parametrised by two functions and a scalar: q_A^*, δ_A^{EMD} and c. It is molded to produce realisations $\hat{q}$ that as a whole track q_A^*, with more variability between realisations at points Φ where $\delta_A^{\mathrm{EMD}}(\Phi)$ is larger (middle panel of Fig. 5.5**a**).

To the best of our knowledge the current literature does not provide a process satisfying all of these constraints. To remedy this situation, we propose a new *hierarchical beta (HB) process*, which we illustrate in Fig. 5.5. A few example realisations of $\hat{q} \sim \mathfrak{Q}_A$ are drawn as grey lines in Fig. 5.5**a**. The mixed PPF q_A^* equation (5.26) is drawn as a green line, while the region corresponding to $q_A^*(\Phi) \pm \sqrt{c}\delta_A^{\mathrm{EMD}}(\Phi)$ is shaded in yellow. The process is well-defined for $c \in [0,\infty]$: the $c \to 0$ limit is simply q_A^*, while the $c \to \infty$ limit is a process that samples uniformly at every step, irrespective of q_A^* and δ_A^{EMD}. In other words, as c increases, the ability of each realisation $\hat{q}_A$ to track q_A^* is limited by the constraints of monotonicity and non-accumulating increments. This translates to step-like PPFs, as seen in the lower panel of Fig. 5.5**a**.

Figure 5.5**b** shows distributions of $\hat{q}(\Phi)$ at three different values of Φ. The value of $q_A^*(\Phi)$ is indicated by the green vertical bar and agrees well with $\mathbb{E}_a\big[\hat{q}(\Phi)\big]$; the desideratum of 5.26 is therefore satisfied. The scaling of these distributions with δ_A^{EMD} equation (5.27) is however only approximate, which we can see as the yellow shading not having the same width in each panel. This is a result of the tension with the other constraints: the HB process ensures that the monotonicity and integrability constraints are satisfied exactly, but allows deviations in the statistical constraints.

A realisation of an HB process is obtained by a sequence of refinements; we illustrate three such refinements steps in Fig. 5.5**d**. The basic idea is to refine an increment $\Delta\hat{q}_{\Delta\Phi}(\Phi) := \hat{q}(\Phi+\Delta\Phi) - \hat{q}(\Phi)$ into two subincrements $\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi)$ and $\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi+\frac{\Delta\Phi}{2})$, with $\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi) + \Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi+\frac{\Delta\Phi}{2}) = \Delta\hat{q}_{\Delta\Phi}(\Phi)$. To do this, we first determine appropriate parameters α and β, draw x_1 from the corresponding beta distribution and assign

$$\begin{aligned}\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi) &\leftarrow x_1 \Delta\hat{q}_{\Delta\Phi}(\Phi)\,,\\ \Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi+\tfrac{\Delta\Phi}{2}) &\leftarrow x_2 \Delta\hat{q}_{\Delta\Phi}(\Phi)\,,\end{aligned} \tag{5.29}$$

where again $x_2 = (1-x_1)$. Figure 5.5**c** shows distributions for the first (orange) and second (blue) subincrements at the fourth refinement step, where we divide an increment over an interval of length $\Delta\Phi = 2^{-3}$ to two subincrements over intervals of length 2^{-4}. Each pair of subincrements is drawn for a different distribution, which depends on the particular realisation $\hat{q}$, but we can nevertheless see the PPF reflected in the aggregate distribution: the PPF has positive curvature, so the second subincrement tends to be larger than the first. Also both increments are bounded from below by 0, to ensure monotonicity.

A complete description of the HB process, including a procedure for choosing the beta parameters α and β such that our desiderata are satisfied, is given in the Methods.

5.2.6.2. Using $\mathfrak{Q}$ to compare models

Having constructed a process $\mathfrak{Q}_{A;c}$ for a candidate model $\mathcal{M}_A$, we can use it to induce a distribution on risks. We do this by generating a sequence of PPFs $\hat{q}_{A,1}, \hat{q}_{A,2}, \ldots, \hat{q}_{A,M_A}$, where $M_A \in \mathbb{N}$ and each $\hat{q}_{A,i}$ is drawn from $\mathfrak{Q}_{A;c}$ (see Fig. 5.5 for examples of sampled $\hat{q}_{A,i}$, and the Methods for more details on how we evaluate B^{EMD}). As we explain in the next section, the sensitivity parameter c is a property of the experiment; it is the same for all candidate models.

For each generated PPF, we evaluate the risk functional (using the integral form of equation (5.20)), thus obtaining a sequence of scalars $R[\hat{q}_{A,1}], R[\hat{q}_{A,2}], \ldots, R[\hat{q}_{A,M_A}]$ that follows $p(R_A \mid \mathfrak{Q}_A)$. With M_A sufficiently large, these samples accurately characterize the distribution $p(R_A \mid \mathfrak{Q}_A)$ (we use $\leftrightarrow$ to relate equivalent descriptions):

$$\begin{aligned} R_A \sim p(R_A \mid \mathcal{D}_{\text{test}}, \mathcal{M}_A; c) \;&\leftrightarrow\; \left\{ R[\hat{q}_{A,1}], R[\hat{q}_{A,2}], \ldots, R[\hat{q}_{A,M_A}] \;\middle|\; \hat{q}_A \sim \mathfrak{Q}_A \right\} \\ &\leftrightarrow\; \left\{ R_{A,1}\,,\;\; R_{A,2}\,,\;\; \ldots, R_{A,M_A} \right\}. \end{aligned} \tag{5.30}$$

Repeating this procedure for a different model $\mathcal{M}_B$ yields a different distribution for the risk:

$$R_B \sim p(R_B \mid \mathcal{D}_{\text{test}}, \mathcal{M}_B; c) \;\leftrightarrow\; \left\{ R_{B,1}\,, R_{B,2}\,, \ldots, R_{B,M_B} \right\}. \tag{5.31}$$

The $B^{\text{EMD}}_{AB;c}$ criterion (5.16) then reduces to a double sum:

$$\begin{aligned} B^{\text{EMD}}_{AB;c} &:= P(R_A < R_B \mid c) \\ &\approx \frac{1}{M_A M_B} \sum_{i=1}^{M_A} \sum_{j=1}^{M_B} 1_{R_{A,i} < R_{B,j}}\,. \end{aligned} \tag{5.32}$$

In equation (5.32), the term within the sum is one when $R_{A,i} < R_{B,j}$ and zero otherwise. A value of $B^{\text{EMD}}_{AB;c}$ greater (less) than 0.5 indicates evidence for (against) model $\mathcal{M}_A$.

The undetermined parameter c (which converts discrepancy into epistemic uncertainty; c.f. 5.27) can be viewed as a way to adjust the sensitivity of the criterion: larger values of c will typically lead to broader distributions of R_A, and therefore lead to a more conservative criterion (i.e. one which is more likely to result in an equivocal outcome). We give some guidelines on choosing c in the next section.

5.2.7. Calibrating and validating the B^{EMD}

The process $\mathfrak{Q}$ constructed in the previous section succeeds in producing distributions of the risk. And at least with some values of the scaling constant c, the distributions look as we expect (Fig. 5.3) and define a B^{EMD} criterion that assigns reasonable tail probabilities (Table 5.1).

To now put the B^{EMD} criterion on firmer footing, recall that it is meant to model variations between data replications. We can therefore validate the criterion by simulating such variations *in silico* (in effect simulating many replications of the experiment): since the true data-generating process $\mathcal{M}_{\text{true}}$ is then known, this gives us a ground truth of known comparison outcomes, against which we can test the probabilistic prediction made by B^{EMD}. Because this procedure will also serve to select a suitable value for the scaling constant c, we call it a *calibration experiment*.

To generate the required variations we use *epistemic distributions* of the type introduced at the top of our Results, of which we already gave an example in equation (5.13). With this we define an alternative tail probability (see the Methods for details),

$$B^{\text{epis}}_{AB;\Omega} := P(R_A < R_B \mid \Omega)\,, \tag{5.33}$$

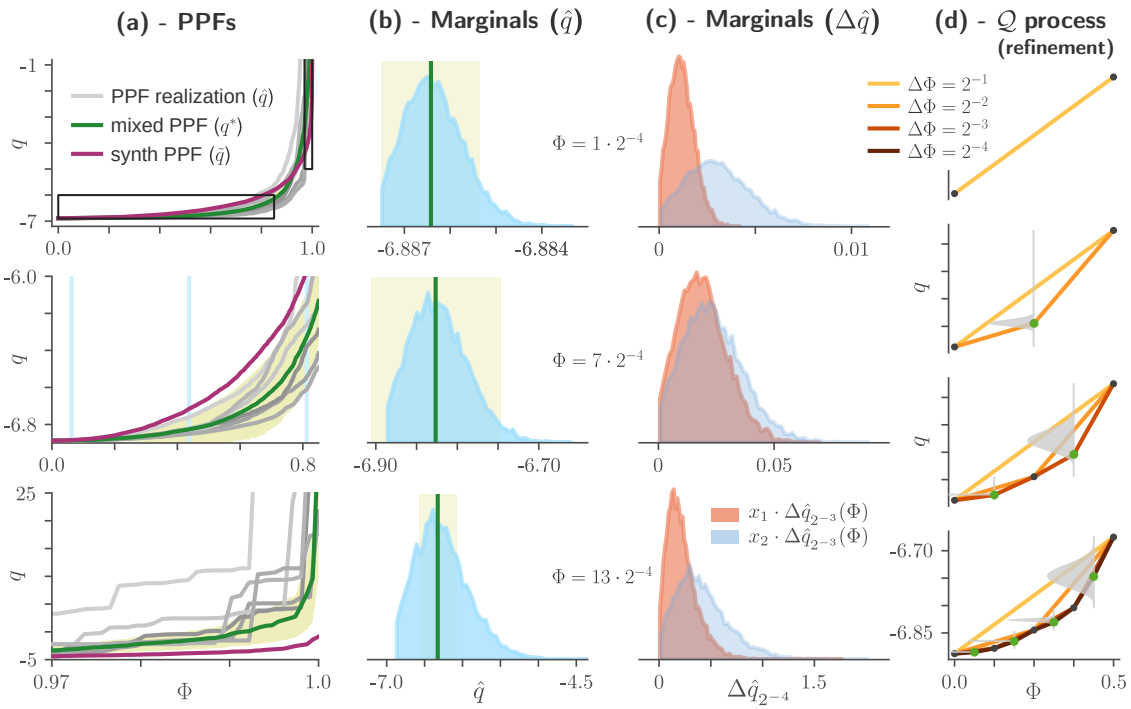


Figure 5.5.: Sampling a hierarchical beta (HB) process. (a) PPF samples (grey lines) drawn from a hierarchical beta process $\mathfrak{Q}$. Mixed (green) and synthetic (red) PPFs are those for the Planck model of Fig. 5.7f (respectively q^* from Equation (5.19) and $\tilde{q}$ from Equation (5.22)). Lower panels are enlarged portions of the top panel, corresponding to the black rectangles in the latter. At each Φ, the variance between realisations is controlled by $\sqrt{c}\,\delta^{\mathrm{EMD}}$ (yellow shading), per 5.27; here we use $c = 0.5$. **(b)** Marginals of $\mathfrak{Q}$ at three values of Φ, obtained as histograms of 10,000 realisations $\hat{q}$. As in (a), the green line indicates the value of $q^*(\Phi)$ and the yellow shading describes the range $q^*(\Phi) \pm \sqrt{c}\delta^{\mathrm{EMD}}(\Phi)$. Values of Φ are given alongside the panels and drawn as cyan vertical lines in (a). **(c)** Distributions of the subincrements drawn at the same three Φ positions as in (b), obtained as histograms from the same 10,000 realisations. Subincrements are for the fourth refinement step, corresponding to the fourth panel of (d). Notation in the legend corresponds to equation (5.29). **(d)** Illustration of the refinement process. This $\mathfrak{Q}$ is parametrised by the same PPFs as the one in (a–c), but we used a larger c (16) to facilitate visualisation. Each refinement step halves the width $\Delta\Phi$ of an increment; here we show four refinements steps, while in most calculations we use eight. New points at each steps are coloured green; the beta distribution from which each new point is drawn is shown in grey. The domain of each of these distributions spans the vertical distance between the two neighbouring points and is shown as a grey vertical line.

which uses the epistemic distribution Ω instead of the process $\mathfrak{Q}_A$ to represent epistemic uncertainty. We then look for $c > 0$ such that the criterion $B^{\mathrm{EMD}}_{AB;c}$ a) is correlated with $B^{\mathrm{epis}}_{AB;\Omega}$; and b) satisfies

$$\left|B^{\mathrm{EMD}}_{AB;c} - 0.5\right| \lesssim \left|B^{\mathrm{epis}}_{AB;\Omega} - 0.5\right|. \tag{5.34}$$

Equation (5.34) encodes a desire for a *conservative criterion*: we are most worried about incorrectly rejecting a model, i.e. of making a type I error. The consequences for a type II error (failing to reject either model) are in general less dire: it is an indication that we need better data to differentiate between models. This desire for a conservative criterion reflects a broad belief in science that it is better to overestimate errors than underestimate them. Given the ontological differences between B^{EMD} and B^{epis} however, in practice one will need to allow at least small violations, which is why we give the relation in equation (5.34) as $\lesssim$.

The advantage of a probability like $B^{\mathrm{epis}}_{AB;\Omega}$ is that it makes explicit which kinds of replication variations are involved in computing the tail probability; the interpretation of equation (5.33) is thus clear. However it can only be computed after generating a large number of synthetic datasets, which requires a known and parametrisable data-generating process $\mathcal{M}_{\mathrm{true}}$; on its own therefore, $B^{\mathrm{epis}}_{AB;\Omega}$ cannot be used directly as a criterion for comparing models. *By choosing c such that the criteria are correlated, we transfer the interpretability of $B^{\mathrm{epis}}_{AB;\Omega}$ onto $B^{\mathrm{EMD}}_{AB;c}$.* Moreover, if we can find such a c, then we have de facto validated $B^{\mathrm{EMD}}_{AB;c}$ for the set of simulated experiments described by Ω.

Since defining an epistemic distribution involves making many arbitrary choices, one may want to define multiple

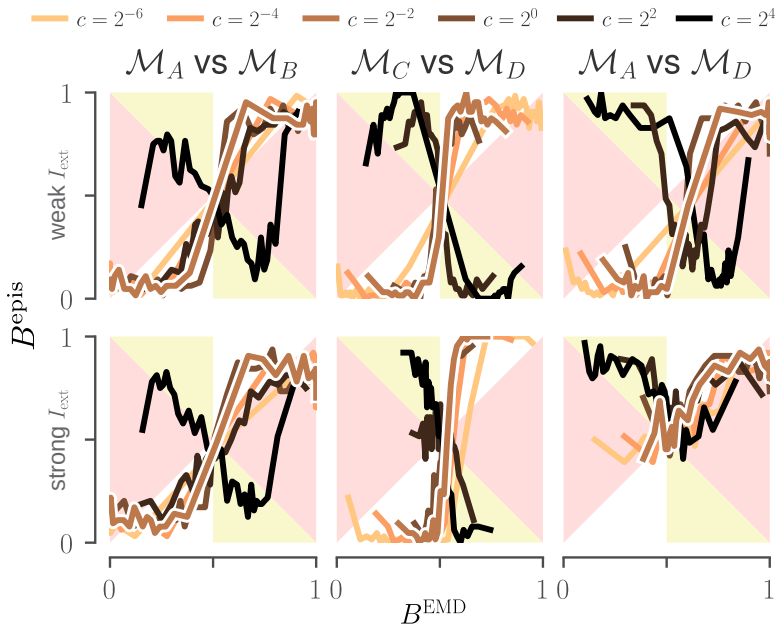


Figure 5.6.: Calibration curves for the LP models of Fig. 5.2. Calibration curves for six epistemic distributions, computed following our proposed calibration procedure. Each curve summarizes 2048 simulated experiments with datasets of size 4000, and the six panels explore how curves depend on three parameters: the pair of models being compared, the constant c and the input strength I_{ext}. The latter is either `weak` **(top)** or `strong` **(bottom)**. Other parameters are kept fixed, namely a `short` correlation time τ and a `Gaussian` observation noise with `low` standard deviation. The regions depicted in red and yellow are those where equation (5.34) is violated. (For an extended version where all conditions are tested, see Supplementary Fig. 5.10. For example R-distributions for each of these c values, see Supplementary Fig. 5.11.)

distributions $\Omega_1, \Omega_2, \ldots$ to ensure that results are not sensitive to a particular choice of Ω. Equation (5.34) can easily be generalised to account for this, following a similar logic as equation (5.15), in which case it becomes

$$\left|B^{\text{EMD}}_{AB;c} - 0.5\right| \lesssim \min_{\Omega \in \{\Omega_1, \Omega_2, \ldots\}} \left|B^{\text{epis}}_{AB;\Omega} - 0.5\right|. \tag{5.35}$$

We found that an effective way to verify equation (5.34) is by plotting $B^{\text{epis}}_{AB;\Omega}$ against $B^{\text{EMD}}_{AB;c}$, where values of $B^{\text{epis}}_{AB;\Omega}$ are obtained by averaging comparison outcomes, conditioned on the value of $B^{\text{EMD}}_{AB;c}$ being within an interval (this is explained more precisely in the Methods). We thus obtain a histogram of $B^{\text{epis}}_{AB;\Omega}$ against $B^{\text{EMD}}_{AB;c}$, which works best when the size of bins is adjusted so that they have similar statistical power. We illustrate this in Fig. 5.6, where histograms are shown as curves to facilitate interpretation. These curves are drawn against the "overconfident regions" (where equation (5.34) is violated), depicted in red or yellow: we look therefore for values of c that stay within the white regions as much as possible. The visual representation makes it easier to judge the extent to which small violations of equation (5.34) can be tolerated. An extended version of this figure where all 48 conditions are tested, is provided as Supplementary Fig. 5.10.

Figure 5.6 shows calibration curves for six different epistemic distributions: three model pairs ($\mathcal{M}_A$ vs $\mathcal{M}_B$, $\mathcal{M}_C$ vs $\mathcal{M}_D$, and $\mathcal{M}_A$ vs $\mathcal{M}_D$), each with `weak` and `strong` external input I_{ext}. (For full details on the choice of epistemic distributions for these experiments, see the Methods.) The model pairs were chosen to test three different situations: one where the candidate models are similar both to each other and the observations ($\mathcal{M}_A$ vs $\mathcal{M}_B$), one where the candidate models are similar to each other but different from the observations ($\mathcal{M}_C$ vs $\mathcal{M}_D$), and one where only one of the candidates is similar to the observations ($\mathcal{M}_A$ vs $\mathcal{M}_D$).

As one might expect, we see some violations of equation (5.35) in Fig. 5.6, which can partly be explained by our choice of a pointwise loss: although pedagogical, it tends to prefer models that produce fewer spikes, as we explain in the Methods. This is partly why, for these types of models, practitioners often prefer loss functions based on domain-specific features like the number of spikes or the presence of bursts [24]. We are also less concerned with the calibration of $\mathcal{M}_A$ vs $\mathcal{M}_D$, since those models are very different (c.f. Fig. 5.2). Not only is the B^{EMD} not needed to

reject $\mathcal{M}_D$ in favour of $\mathcal{M}_A$, but the variational approximation expressed by 5.27 works best when the discrepancy between the model and the observed data (i.e. δ^{EMD}) is not too large.

Nevertheless, for values of c between 2^{-4} and 2^0, we see that calibration curves largely avoid the overconfidence (red and yellow) regions under most conditions and that B^{epis} is strongly correlated with B^{EMD}. This confirms that $B^{\mathrm{EMD}}_{ab;c}$ can be an estimator for the probability $P(R_a < R_b)$ (recall Equations (5.16) and (5.33)) and validates our choice of $c = 2^{-2}$ for the R-distributions in Fig. 5.2. More importantly, it shows that c does not need to be tuned to a specific value for B^{EMD} to be interpretable: it suffices for c to be within a finite range. We observe a similar region of validity for c with the model introduced in the next section (see Fig. 5.9 in the Methods). The fact that B^{EMD} remains valid for a range of c values is a major reason why we think it can be useful for analysing real data, where we don't know the epistemic distribution, as well as for comparing multiple models simultaneously.

Within its range of validity, the effect of c is somewhat analogous to a confidence level: just like different confidence levels will lead to different confidence intervals, different values of c can lead to different (but equally valid) values of $B^{\mathrm{EMD}}_{ab;c}$. See the Supplementary Discussion for more details. Note also that due to the constraints of the $\mathfrak{Q}$ process, increasing c does not simply increase the spread of R-distributions, but can also affect their shape; this is illustrated in Supplementary Fig. 5.11.

There are limits however to the range of validity. Too small values of c will remove any overlap between R-distributions and produce an overconfident B^{EMD}. Too large values of c will exaggerate overlaps, underestimating statistical power. Large values can also lead to distortions in the $\mathfrak{Q}$ process, due to the monotonicity constraint placing an upper bound on the achievable metric variance; we see this as the curves reversing in Fig. 5.6 for $c \geq 2^2$. In the Supplementary Discussion we list some possible approaches to enlarge the range of validity, or otherwise improve the calibration of the B^{EMD}.

5.2.8. Characterizing the behaviour of R-distributions

To better anchor the interpretability of R-distributions, in this section we perform a more systematic study of the relationship between epistemic uncertainty, aleatoric uncertainty, and the shape of the R-distributions. To do this we use a different example, chosen for its illustrative simplicity, which allows us to independently adjust the ambiguity (how much two models are qualitatively similar) and the level of observation noise.

Concretely, we imagine a fictitious historical scenario where the Rayleigh-Jeans

$$\mathcal{B}_{\mathrm{RJ}}(\lambda; T) = \frac{2ck_BT}{\lambda^4} \tag{5.36}$$

and Planck

$$\mathcal{B}_{\mathrm{P}}(\lambda; T) = \frac{2hc^2}{\lambda^5} \frac{1}{\exp\left(\frac{hc}{\lambda k_B T}\right) - 1} \tag{5.37}$$

models for the radiance of a black body are two candidate models given equal weight in the scientific community. They stem from different theories of statistical physics, but both agree with observations at infrared or longer wavelengths, and so both are plausible if observations are limited to that window. (When it is extended to shorter wavelengths, the predictions diverge and it becomes clear that the Planck model is the correct one.)

Remark 5.2.1 For our purposes, these are just two models describing the relationship between an independent variable λ (the wavelength) and a dependent variable $\mathcal{B}$ (the spectral radiance), given a parameter T (the temperature) that is inferred from data; our discussion is agnostic to the underlying physics. The parameters h, c and k_B are known physical constants (the Planck constant, the speed of light and the Boltzmann constant) and can be omitted from the discussion.

We use a simple Poisson counting process to model the data-generating model $\mathcal{M}_{\mathrm{true}}$ including the observation noise:

$$\mathcal{B} \mid \lambda, T, s \sim \frac{1}{s} \mathrm{Poisson}\big(s\, \mathcal{B}_{\mathrm{P}}(\lambda; T)\big) + \mathcal{B}_0\,, \tag{5.38}$$

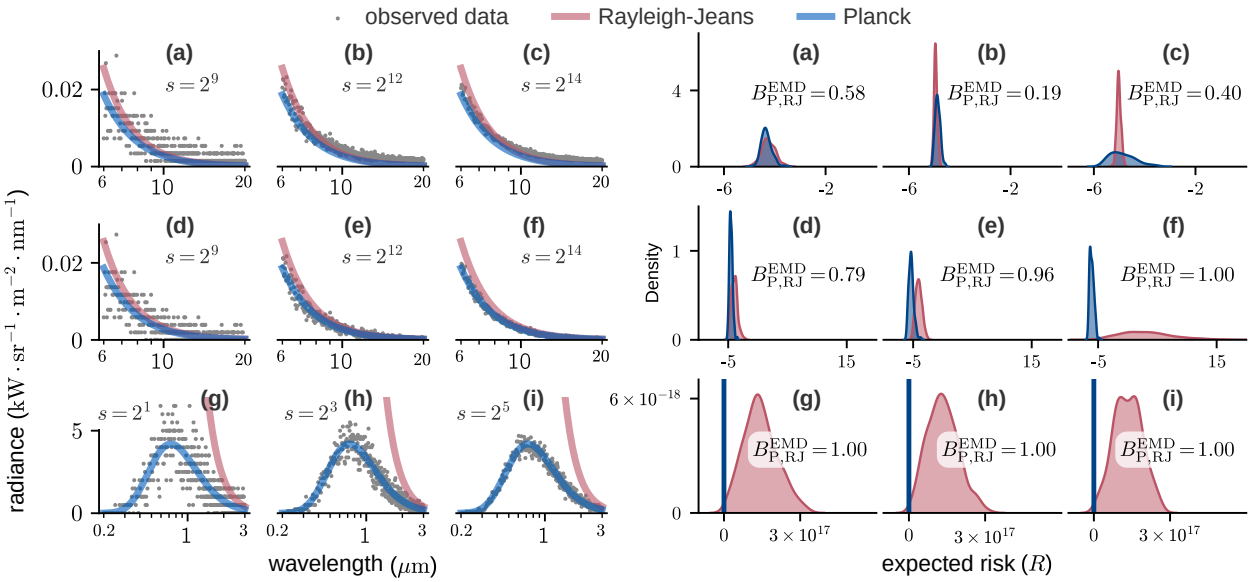


Figure 5.7.: *R*-**distributions** (right) **for different simulated datasets of spectral radiance** (left). Datasets were generated using equation (5.38). For each model a, the R_a-distribution was obtained by sampling an HB process $\mathfrak{Q}$ parametrised by δ_a^{EMD} and a sensitivity $c = 2^{-1}$; see the respective Results sections for definitions of $\mathfrak{Q}$ and δ^{EMD}. A kernel density estimate is used to visualise the resulting samples equation (5.30) as densities. In all rows, noise (s) decreases left to right. **Top row:** Over a range of long wavelengths with positive bias $\mathcal{B}_0 = 0.0015$, both models fit the data equally well. **Middle row:** Same noise levels as the first row, but now the bias is zero. Planck model is now very close to $\mathcal{M}_{\mathrm{true}}$, and consequently has nearly-Dirac R-distributions and lower expected risk. **Bottom row:** At visible wavelengths, the better fit of the Planck model is incontrovertible.

where s is a parameter related to the gain of the detector (see the Methods for details). Most relevant to the subsequent discussion is that the mean and variance of $\mathcal{B}$ are

$$\mathbb{E}[\mathcal{B}] = \mathcal{B}_{\mathrm{P}}(\lambda; T) + \mathcal{B}_0\,,$$
$$\mathbb{V}[\mathcal{B}] = \frac{\mathcal{B}_{\mathrm{P}}(\lambda; T)}{s}\,,$$

and can therefore be independently controlled with the parameters $\mathcal{B}_0$ and s.

For the purposes of this example, both *candidate* models $\mathcal{M}_{\mathrm{RJ}}$ and $\mathcal{M}_{\mathrm{P}}$ make the incorrect (but common) assumption of additive Gaussian noise, such that instead of equation (5.38) they assume

$$\mathcal{B} \mid \lambda, T, \sigma \sim \mathcal{N}\big(\mathcal{B}_a(\lambda; T), \sigma^2\big)\,, \tag{5.39}$$

with $\mathcal{B}_a \in \{\mathcal{B}_{\mathrm{RJ}}, \mathcal{B}_{\mathrm{P}}\}$ and $\sigma > 0$. This ensures that there is always some amount of mismatch between $\mathcal{M}_{\mathrm{true}}$ and the two candidates. That mismatch is increased when $\mathcal{B}_0 > 0$, which we interpret as a sensor bias that the candidate models neglect.

With this setup, we have four parameters that move the problem along three different "axes": The parameters λ_{min} and λ_{max} determine the spectrometer's detection window, and thereby the **ambiguity**: the shorter the wavelength, the easier it is to distinguish the two models. The parameter s determines the **level of noise**. The parameter $\mathcal{B}_0$ determines an additional amount of **misspecification** between the candidate model and the data.

We explore these three axes in Fig. 5.7, and illustrate how the overlap of the R_{P} and R_{RJ} distributions changes through mainly two mechanisms: Better data can shift one R-distribution more than the other, and/or it can tighten one or both of the R-distributions. Either of these effects can increase the separability of the two distributions (and therefore the strength of the evidence for rejecting one of them).

5.2.9. Different criteria embody different notions of robustness

As a final step, we now wish to locate our proposed method within the wider constellation of model selection methods.

The motivation behind any model selection criterion is to make the selection in some way robust—i.e. to encourage a choice that is good not just for the given particular data and models, but also for variations thereof. Otherwise we would just select the model with the lowest empirical risk and be done with it. Criteria vary widely however in terms of what kinds of variations they account for. For the purposes of comparing with the EMD rejection rule, we consider three types:

In-distribution variations of the data where new samples are drawn from the *same* data-generating process $\mathcal{M}_{\text{true}}$ as the data.

Out-of-distribution variations of the data where new samples are drawn from a *different* data-generating process $\mathcal{M}'_{\text{true}}$ (than the data-generating process $\mathcal{M}_{\text{true}}$). How much $\mathcal{M}'_{\text{true}}$ is allowed to differ from $\mathcal{M}_{\text{true}}$ will depend on the target application.

Variations of model parameters, for example by sampling from a Bayesian posterior, or refitting the model to new data.

Robustness to data variations, whether in- or out-of-distribution, is generally considered a positive. In contrast, robustness to model variations (what might more commonly be described as *insensitivity* to model parameters) is often attributed to excess model complexity and thus considered a negative.

Exactly which type of variation a criterion accounts for—and just as importantly, *how* it does so—defines what we might call the *paradigm* for that criterion. In Fig. 5.8 we compare the EMD approach to five other commonly used criteria with a variety of model selection paradigms; we expand on those differences below.

For our purposes the most important of those differences is how a criterion accounts for epistemic uncertainty. Four of the considered criteria do this by comparing parametrised *families* of models—either explicitly by averaging their performance over a prior (BIC, $\log \mathcal{E}$), or implicitly by refitting the model to each new dataset (MDL, AIC). In other words, such criteria compare the mathematical structure of different models, rather than specific parameters; they would not be suited to the neural circuit example of earlier sections, where the candidate models were solely distinguished by their parameters. The comparison we do here is therefore *only possible* when the models being compared are *structurally distinct*.

Our comparison is also tailored to highlight features of particular importance for experimental data and which differentiate our method. It is by no means exhaustive, and many aspects of model comparison are omitted—such as their consistency or how they account for informative priors. The example data were also chosen to be in a regime favourable to all models; this is partly why many panels show similar behaviour.

We can broadly classify criteria according to whether they select models based solely on their *predictive accuracy*, or whether they also penalize their *complexity* (so as to prefer simpler models).

Let us begin with the latter. The most common of these is likely the *Bayes factor*, formally defined as the ratio of model probabilities and in practise the ratio of marginal likelihoods, or *model evidence* [59, 84, 134]:

$$B^{\text{Bayes}}_{AB} = \frac{p(\mathcal{M}_A \mid \mathcal{D})}{p(\mathcal{M}_B \mid \mathcal{D})} = \frac{\int p(\mathcal{D} \mid \theta, \mathcal{M}_A)\pi_A(\theta)d\theta}{\int p(\mathcal{D} \mid \theta, \mathcal{M}_B)\pi_B(\theta)d\theta} =: \frac{\mathcal{E}_A}{\mathcal{E}_B}\,. \tag{5.40}$$

(We previously defined the evidence $\mathcal{E}_A$ the same way in equation (5.6).) Here $p(\mathcal{D} \mid \theta, \mathcal{M}_A)$ and π_A are likelihood and prior distributions respectively. Going forward, instead of Bayes factors we will consider each model's log evidence: the former are easily recovered from the difference of the latter,

$$\log B^{\text{Bayes}}_{AB} = \log \mathcal{E}_A - \log \mathcal{E}_B\,, \tag{5.41}$$

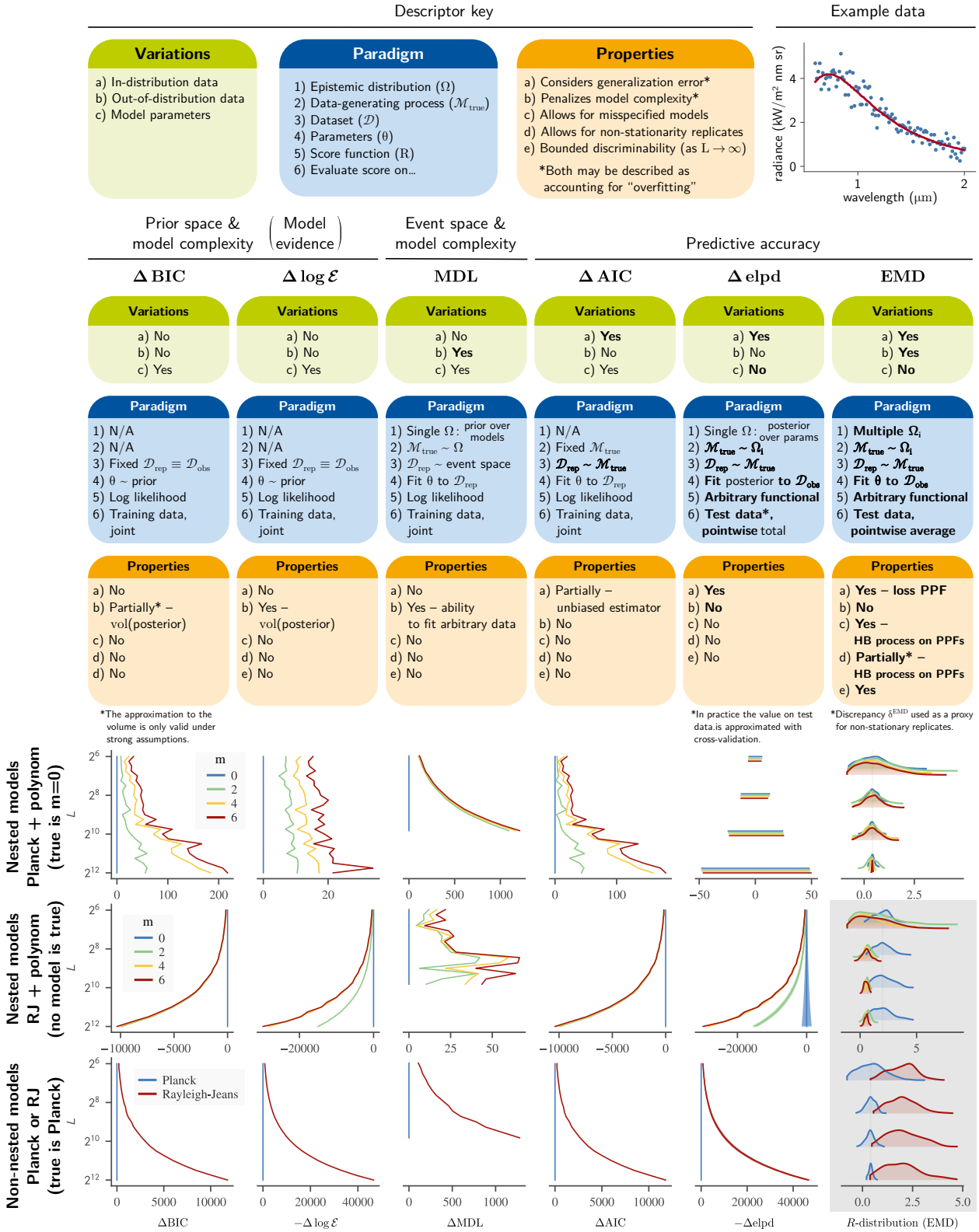


Figure 5.8.: Behaviour of model selection criteria as a function of dataset size Datasets were generated using equation (5.38); they differ only in their number of samples and random seed; an example is shown in the top right. The comparison outcome between two models is given by the difference between their criteria; these are plotted as differences w.r.t. an arbitrary reference model (blue curve). The log evidence ($\log \mathcal{E}$) and elpd provide uncertainties, shown either as shading or as error bars. MDL curves are shorter because computations exceeding 100 hours were aborted.

and this makes it easier to compare multiple models at once. It also matches the conventional format of information criteria. Numerical evaluation of the model evidence is non-trivial, but generic software solutions are available. For this comparison we used an implementation of nested sampling [80] provided by the Python package `dynesty` [170].

The model evidence corresponds to studying *variations of the model* (represented by the prior) for a *fixed dataset* $\mathcal{D}$. A model is penalised if its prior is unnecessarily broad or high-dimensional; this is the so-called "Occam's razor" principle that favours simpler models [59, 84] . This sometimes leads to undesirable results when priors are non-informative [79].

The *Bayesian information criterion* (BIC) is an asymptotic approximation of the log evidence, either around the maximum likelihood or maximum a posteriori estimate [86, 134]. It will approach the log evidence ($\log \mathcal{E}$) when the likelihood is non-singular and approximately Gaussian around its maximum.

Another approach that penalizes complexity is the *minimum description length (MDL)*. This is more accurately described as a methodological framework built around the choice of a "universal probability distribution" [92]; usually however the *normalised maximum likelihood* (NML) distribution is implied, and this is what we do here. Assuming equal prior preference for each model, MDL selects the one with the highest NML probability; equivalently, it computes for each model the quantity

$$\mathrm{MDL}_A := -\max_\theta p(\mathcal{D} \mid \mathcal{M}_A(\theta)) + \mathrm{COMP}(\mathcal{M}_A)\,. \tag{5.42}$$

and selects the model for which MDL_A is minimal. (It is assumed here that $\mathcal{M}_A$ is parametrised, and $\mathcal{M}_A(\theta)$ denotes the model $\mathcal{M}_A$ with its parameters set to the values θ.) The first term is the maximum of the likelihood evaluated on the training data. The second term is the *MDL complexity*, defined by integrating over the entire event space:

$$\mathrm{COMP}(\mathcal{M}_A) := \int \max_\theta p(\mathcal{D}' \mid \mathcal{M}_A(\theta))\, d\mathcal{D}'\,. \tag{5.43}$$

In other words, MDL complexity quantifies the ability of a model to fit arbitrary data. Note that it does not matter here what data-generating process $\mathcal{M}_\text{true}$ generated the data since each possible dataset is given equal weight. Approximations are almost always required to compute equation (5.43) since the integral is usually intractable; for this comparison, we chose a data regime where it is just about possible to enumerate the possible datasets and estimate $\mathrm{COMP}(\mathcal{M}_A)$ directly with Monte Carlo.

With MDL therefore we consider variations of the models, since each model is refitted to each dataset $\mathcal{D}'$. We have no in-distribution dataset variations (the true process $\mathcal{M}_\text{true}$ is ignored), but we do have out-of-distribution variations in the form of the integral over the event space.

Moving on now to criteria that compare predictive accuracy, we start with the *expected log pointwise predictive density (elpd)* [56]. This is defined as the expectation of the loss Q for L samples drawn from the data-generating process:

$$\begin{aligned}\mathrm{elpd}_A &:= \sum_{i=1}^{L} \mathbb{E}_{(x_i,y_i)\sim\mathcal{M}_\text{true}}\big[Q(x_i, y_i; \mathcal{M}_A)\big] \\ &= L\, R_A\,.\end{aligned} \tag{5.44}$$

The most common form of the elpd uses the (negative) log likelihood as the loss function Q. In this form where each sample is drawn from the same distribution, the elpd is proportional to the risk R_A defined in equation (5.4), and therefore conceptually analogous: it describes robustness to in-distribution variations of the dataset. The models are kept fixed in these comparisons; they are not refitted to the new data. The uncertainty on elpd comes from the finite number of samples used to compute the expectation.

Since in current practice the elpd is mostly used with Bayesian models, this is also what we do in Fig. 5.8. Efficient estimators exist for computing the elpd from training data, using either leave-one-out cross-validation [56] or the widely applicable information criterion (WAIC) [171]. This avoids the need to reserve a separate test dataset. We used `ArviZ`'s [78] implementation of WAIC for this comparison.

In contrast to the elpd, the *Akaike information criterion (AIC)* does not assume a fixed model. Although it also ranks models according to their expected loss, this time the expectation is (conceptually) over both the test set $\mathcal{D}_{\text{test}}$ and training data $\mathcal{D}_{\text{train}}$, both drawn independently from the data-generating process [160, 161]:

$$\operatorname{elpd}_A^{\text{AIC}} = \mathbb{E}_{\mathcal{D}_{\text{test}}\sim\mathcal{M}_{\text{true}}}\mathbb{E}_{\mathcal{D}_{\text{train}}\sim\mathcal{M}_{\text{true}}}\left[Q(\mathcal{D}_{\text{test}}; \mathcal{M}_A(\hat{\theta}_{\mathcal{D}_{\text{train}}}))\right], \tag{5.45}$$

where $\hat{\theta}_{\mathcal{D}_{\text{train}}}$ is the maximum likelihood estimate given the training data. Concretely however the AIC is a simple statistic, similar to the BIC and other information criteria:

$$\underset{A}{\text{AIC}} := 2\,Q(\mathcal{D}_{\text{train}}; \mathcal{M}_A(\hat{\theta}(\mathcal{D}_{\text{train}})) + 2k_A\,, \tag{5.46}$$

where k_A is the number of parameters of model $\mathcal{M}_A$. The derivation of the AIC relies on Q being the log likelihood, and assumes that the likelihood is approximately Gaussian and non-singular around $\hat{\theta}_{\mathcal{D}_{\text{train}}}$. When these conditions are verified, then $\Delta\,\text{AIC}_{AB} := \text{AIC}_A - \text{AIC}_B$ is an unbiased estimator of the difference $\operatorname{elpd}_A^{\text{AIC}} - \operatorname{elpd}_B^{\text{AIC}}$. In this case, the model with the lowest AIC will—in expectation—also be the one with the lowest expected risk.

So how does the B^{EMD} compare to the criteria listed above? First, it computes R-distributions for *fixed models*. This is a key feature, since otherwise it is impossible to compare different parameter vectors for otherwise structurally identical models. (Note that EMD also allows models to have different structure—it simply does not require it.) This also means that the B^{EMD} does not penalise model complexity, in contrast to the evidence, MDL and AIC.

Second, like the elpd, the B^{EMD} accounts for *in-distribution variability* by estimating the risk using held-out test data $\mathcal{D}_{\text{test}}$.

Third, EMD R-distributions also account for *out-of-distribution variability*. Conceptually this is done via the epistemic distributions Ω, concretely via the HB process on PPFs—the variance of the latter stemming ultimately from the discrepancy function δ^{EMD}. Better models therefore result in less variability, reflecting our better grasp of the process being modelled. In contrast, only the MDL allows for some form of out-of-distribution variability (in the form of a complexity penalty), and it is essentially rigid: always an integral over the entire event space.

The biggest difference between the EMD R-distributions and other criteria however is their behaviour as the number of samples L increases. This we illustrate in the lower part of Fig. 5.8, which plots the value of a criterion for different models as a function of L. We do this for three collections of models, where the data-generating process is always the Planck model $\mathcal{B}_{\text{P}}$ described in the previous section. In all panels, scores are defined so that models with lower scores are preferred.

In the first row we compare four candidate models of the form

$$\mathcal{B} \mid \lambda, T, \sigma, \{b_j\} \sim \mathcal{N}\left(\mathcal{B}_{\text{P}}(\lambda; T) + \sum_{j=0}^{m} b_j \lambda^j,\ \sigma^2 \mathcal{B}_{\text{P}}(\lambda; T)\right). \tag{5.47}$$

The candidates are therefore *nested*: any model within the class $\mathcal{M}_{m=2}$ can recovered from the class $\mathcal{M}_{m=4}$ by fixing $b_3 = b_4 = 0$. Moreover, all models effectively contain $\mathcal{M}_{\text{true}}$, since it corresponds to $m = 0$. (The data are in a regime where the Poisson distribution of $\mathcal{M}_{\text{true}}$ is close to Gaussian.)

The second row is similar, but expands around the Rayleigh-Jeans model instead:

$$\mathcal{B} \mid \lambda, T, \sigma, \{b_j\} \sim \mathcal{N}\left(\mathcal{B}_{\text{RJ}}(\lambda; T) + \sum_{j=0}^{m} b_j \lambda^j,\ \sigma^2 \mathcal{B}_{\text{RJ}}(\lambda; T)\right). \tag{5.48}$$

In this case all models are misspecified, but those with higher degree polynomials are better able to compensate for this and thus achieve lower prediction errors.

The third row compares the two $m = 0$ models from the first two rows.

If we interpret the score difference between two models (measured as the horizontal distance between two curves) as the strength of the evidence supporting a particular model choice, then it is clear from Fig. 5.8 that for all criteria

except EMD, strength grows unbounded as we increase the number of samples L. A bounded strength however is desirable when analyzing experimental data, since no experiment has infinite accuracy.

In contrast, we see that the R-distributions in the grey shaded area stabilize as a function of L. Therefore also the tail probabilities $B^{\mathrm{EMD}}_{ab;c} = P(R_A < R_B \ : \ c)$ (equation (5.16)), which define the EMD rejection rule, converge in general to finite, non-zero values. In some cases the R-distributions may approach a Dirac delta as L increases, as we see in the first row of Fig. 5.8. This occurs when the candidate model is able to match $\mathcal{M}_{\mathrm{true}}$ exactly.

We provide a different set of comparisons in the Supplementary Results, listing numerical values of criteria differences in Supplementary Table 5.3 to complement the graphical presentation of Fig. 5.8. These alternative comparisons place greater emphasis on how misspecification can affect a criterion's value, and how that value can (mis)represent the strength of the evidence.

5.3. Discussion

Model selection in the scientific context poses unique challenges. Ranking models according to their empirical risk captures the scientific principle that models should be able to predict new data, but in order for results to be *reproducible* across experiments, that risk should also be *robust to variations in the replication process*. We propose that the empirical model discrepancy between quantile functions (δ^{EMD})—measured from a single dataset—can serve as a baseline for the uncertainty across replications. We represent this uncertainty by the stochastic process $\mathfrak{Q}$; due to intrinsic constraints of quantile functions, this process is relatively stereotyped, allowing us to automate the generation of *R(isk)-distributions*—visually summarized in Fig. 5.1—from empirical data (see Code availability). Moreover, we can calibrate the process against simulated replications—by adjusting the proportionality factor c that converts model discrepancy into epistemic uncertainty—so that it approximates the epistemic variability expected in experiments. In the end, the decision to reject a model in favour of another reduces to a simple tail probability $B^{\mathrm{EMD}}_{AB;c} = P(R_A < R_B \mid c)$ between two R-distributions (equation (5.12)). Crucially, we do not force a model choice: a model is only rejected if this probability exceeds a chosen threshold ϵ. Guidelines for choosing rejection thresholds should be similar to those for choosing significance levels.

A key feature of our approach is to compare models based on their risk. This is an uncommon choice for statistical model selection, because it tends to select overfitted models when the same data are used both to fit and test them. In more data-rich machine learning paradigms however, risk is the preferred metric, and overfitting is instead avoided by testing models on held-out data. Our method assumes this latter paradigm, in line with our motivation to compare complex, data-driven models. A selection rule based on risk is also easily made consistent (in the sense of asymptotically choosing the correct model) by choosing a proper loss function, such as the negative log likelihood. Moreover, with the specific choice of a log likelihood loss, differences in risk can be interpreted as differences in differential entropy. The risk formulation however is more general, and allows the choice of loss to be tailored to the application.

We illustrated the approach on two example problems, one describing the radiation spectrum of a black body, and the other the dynamical response of a neuron of the lobster pyloric circuit. In the case of the former, we compared the Planck and Rayleigh-Jeans models; these being two structurally different models, we could also compare the B^{EMD} with other common selection criteria (Fig. 5.8). We thereby highlight not only that different criteria account for different types of variations, but also that B^{EMD} probabilities are unique in having well-defined, finite limits (i.e. neither zero nor infinite) when the number of samples becomes large. This is especially relevant in the scientific context, where we want to select models that work for datasets of all sizes. The simplicity of the black body radiation models also allowed us to systematically characterize how model ambiguity, misspecification, and observation noise affect R-distributions (Fig. 5.7).

The neural dynamics example was inspired by the increasingly common practice of data-driven modelling. Indeed, Prinz et al. [15, 21]—whose work served as the basis for this example—can be viewed as early advocates for data-driven approaches. Although their exhaustive search strategy faces important limitations [172], more recent approaches are much more scalable. For example, methods using gradient based optimization can simultaneously learn dozens—or in the case of neural networks, millions—of parameters [96, 173]. These methods however are generally used to solve

non-convex problems, and therefore run against the same follow-up question: *having found (possibly many) candidate models, which ones are truly good solutions, and which ones should be discarded as merely local optima?* The B^{EMD} probabilities, which account for epistemic uncertainty on a model's risk, can address this latter question. They can do so even when candidate models are structurally identical (distinguished therefore only by the values of their parameters), which sets this method apart from most other model selection criteria.

A few other model selection methods have been proposed for structurally identical candidate models, either within the framework of approximate Bayesian computing [174] or by training a machine learning model to perform model selection [175]. Largely these approaches extend Bayesian model selection to cases where the model likelihood is intractable, and as such should behave similarly to the Bayesian methods studied in Fig. 5.8. Epistemic uncertainty is treated simply as a prior over models, meaning in particular that these approaches do not account for variations in the replication process.

Epistemic uncertainty itself is not a novel idea: Kiureghian and Ditlevsen [149] discuss how it should translate into scientific practice, and Hüllermeier and Waegeman [150] do the same for machine learning practice. There again however, the fact that real-world replication involves variations of the data-generating process is left largely unaddressed. Conversely, the property of a model of being robust to replication variations is in fact well ingrained in the practice of machine learning, where it is usually referred to as "generalisability", or more specifically "out-of-distribution performance". This performance however is usually only measured post hoc on specific alternative datasets. Alternatively, there have been efforts to quantify the epistemic uncertainty by way of ensemble models, including the GLUE methodology [152, 176], Bayesian calibration [154, 177], Bayesian neural networks [157], and drop-out regularization at test time [155]. Since these methods focus on quantifying the effect of uncertainty on *model predictions*, they improve the estimate of risk, but still do not assign uncertainty to that estimate. In contrast, with the hierarchical beta process $\mathfrak{Q}$, we express epistemic uncertainty on the *statistics of the sample loss*. This has two immediate advantages: the method trivially generalises to high-dimensional systems, and we obtain *distributions* for the risk (Fig. 5.3). In this way, uncertainty in the model translates to uncertainty in the risk.

A few authors have argued for making robustness against replication uncertainty a logical keystone for translating fitted models into scientific conclusions [147, 148], although without modelling them explicitly. There, as here, the precise definition of what we called an epistemic distribution is left to the practitioner, since appropriate choices will depend on the application. One would of course expect a selection rule not to be too sensitive to the choice of epistemic distribution, exactly because it should be robust.

One approach that does explicitly model replications, and uses similar language as our own to frame its question, is that of Gelman, Meng, and Stern [178]. Those authors also model the distribution of discrepancies (albeit here again of model predictions rather than losses) and compute tail probabilities. Their method however is designed to assess the plausibility of a single model; it would not be suited to choosing the better of two approximations. It is also intrinsically Bayesian, using the learned model itself (i.e. the posterior) to simulate replications; we do not think it could be generalised to non-Bayesian models like our neural circuit example.

Building on similar Bayesian foundations but addressing model comparison, Moran, Cunningham, and Blei [179] propose the posterior predictive null check (PPN), which tests whether the predictions of two models are statistically indistinguishable. In contrast to the B^{EMD}, the PPN is purely a significance test: it can detect if two models are distinguishable, but not which one is better, or by how much. One can see parallels between this approach and the model selection tests (MST) proposed by Golden [74, 180]. While MST is framed within the same paradigm as AIC—and so makes the same approximation that inferred parameters follow a Gaussian distribution—instead of simply proposing an unbiased estimator for the difference $R_A - R_B$, that difference is mapped to a χ^2 distribution. This allows the author to propose a selection procedure similar in spirit to equation (5.11), but where the "no rejection" case is selected on the basis of a significance test. Although replication uncertainty is not treated—and therefore the test can be made arbitrarily significant by increasing the number of samples—the author anticipates our emphasis on misspecification and our choice to select models based on risk.

A guiding principle in designing the EMD rejection rule was to emulate how scientists interpret noisy data. This approach of using conceptually motivated principles to guide the development of a method seems to be fruitful, as it has lead to other recent developments in the field of model inference. For instance, Swigon et al. [57] showed that by requiring a model to be invariant under parameter transformation, one can design a prior (i.e. a regulariser) that better predicts aleatoric uncertainty. Another example is simulation-based calibration (SBC) [135, 181], for which

the principle is self-consistency of the Bayesian model. This is conceptually similar to the calibration procedure we proposed: in both case a consistency equation is constructed by replacing one real dataset with many simulated ones. The main difference is that SBC checks for consistency with the Bayesian prior (and therefore the aleatoric uncertainty), while we check for consistency with one or more epistemic distributions. In both cases the consistency equation is represented as a histogram: in the case of SBC it should be flat, while in our case it should follow the identity $B^{\text{epis}} = B^{\text{EMD}}$.

The model discrepancy δ^{EMD} (equation (5.23)) also has some similarities with the Kolmogorov-Smirnoff (K-S) statistic; the latter is defined as the difference between two CDFs and is also used to compare models. Methodologically, the K-S statistic is obtained by taking the supremum of the difference, whereas we keep δ^{EMD} as a function over $[0, 1]$; moreover, a Kolmogorov-Smirnoff test is usually computed on the distribution of data samples ($\mathcal{D}_{\text{test}}$) rather than that of their losses ($\{Q(x, y) : (x, y) \in \mathcal{D}_{\text{test}}\}$).

One would naturally expect a comparison criterion to be symmetric: the evidence required to reject model $\mathcal{M}_A$ must be the same whether we compare $\mathcal{M}_A$ to $\mathcal{M}_B$ or $\mathcal{M}_B$ to $\mathcal{M}_A$. Moreover, it must be possible that *neither* model is rejected if the evidence is insufficient. Bayes factors and information criteria do allow for this, but with an important caveat: the relative uncertainty on those criteria is strongly tied to dataset size, so much so that with enough samples, there is *always* enough data to reject a model, as we illustrate in Fig. 5.8. This is in clear contradiction with scientific experience, where more data cannot compensate for bad data.

Another important consequence of a symmetric criterion is that it trivially generalises to comparing any number of models, such as we did in Table 5.1. One should however take care in such cases that models are not simply rejected by chance when a large number of models are simultaneously compared. By how much that chance would increase, and how best to account for this, are questions we leave for future work.

From a practical standpoint, the most important feature of the B^{EMD} is likely its ability to compare both specific model parametrisations of the same structural model, as well as structurally different models. Most other criteria listed in Fig. 5.8 only compare model structures, because they either assume globally optimal parameters (AIC, BIC), integrate over the entire parameter space (Bayes factor) or refit their parameters to each dataset (AIC, MDL). The calculation of the $B^{\text{EMD}}_{AB;c}$ between two models is also reasonably fast, taking less than a minute in most of our examples. Although the calibration experiments are more computationally onerous, for both of our models we found that the $B^{\text{EMD}}_{AB;c}$ is valid over a range of c values, with the high end near $c = 1$. This suggests a certain universality to the $B^{\text{EMD}}_{AB;c}$, which would allow practitioners to perform initial analyses with a conservative value like $c = 1$, only then proceeding to calibration experiments if there is indication that they are worth pursuing. Alternatively, since calibration is a function of experimental details, a laboratory might perform it once to determine a good value of c for its experimental setup, and then share that value between its projects.

This property that the same value of c can be used to compare different models in different contexts is the key reason why we expect the B^{EMD} to generalise from simulated epistemic distributions to real-world data. It is clear however that this cannot be true for arbitrary epistemic distributions: we give some generic approaches for improving the outcome of calibrations in the Supplementary Discussion, but the B^{EMD} criterion will always work best in cases where one has a solid experimental control (to minimize variability between replications) and deep domain knowledge (to accurately model both the system and the replications).

Another important feature is that computing the B^{EMD} only requires knowledge that is usually accessible in practice: a method to generate synthetic samples from both $\mathcal{M}_A$ and $\mathcal{M}_B$, a method to generate true samples, and a loss function. Some reasoned choices are used to define the stochastic process $\mathfrak{Q}$ in the PPF space, but they are limited by the hard constraints of that space: PPFs must be one-dimensional on $[0, 1]$, monotone, integrable, and non-accumulating. Although they make defining an appropriate stochastic process considerably more complicated, these constraints seem to be key for obtaining distributions that are representative of epistemic uncertainty (see Figure 5.13). We proposed *hierarchical beta (HB) processes* (further detailed in the Methods) to satisfy these constraints, but it would be an interesting avenue of research to look for alternatives that also satisfy the desiderata for $\mathfrak{Q}$. In particular, if one could define processes that better preserve the proportionality of 5.27 at large values of c, or allow for a faster computational implementation, the B^{EMD} could be applied to an even wider range of problems.

There are other open questions that would be worth clarifying in future work. One is the effect of finite samples: having fewer samples increases the uncertainty on the shape of the estimated mixed PPF q^*, and should therefore

increase the discrepancy with the synthetic PPF $\tilde{q}$ (and thereby the estimated epistemic uncertainty). In this sense our method accounts for this implicitly, but whether it should do so explicitly—and how—is yet unexplored. It may also be desirable in some cases to use the same samples both to fit and test models, as it is commonly done in statistical model selection. One might explore for such cases the possibility of combining our approach with existing methods, such as those developed to estimate the elpd [56, 182], to avoid selecting overfitted models.

Looking more broadly, it is clear that the wide variety of possible epistemic distributions cannot be fully captured by the linear relationship of 5.27, which formalises as a desideratum for the stochastic process $\mathfrak{Q}$ what we referred to as the *EMD principle*. Going forward, it will be important therefore to consider the B^{EMD} criterion an imperfect solution, and to continue looking for ways to better account for epistemic uncertainty. These could include relating misspecification directly to the distribution of losses (going further than the B^{Q} considered our Supplementary Discussion), or more sophisticated self-consistency metrics than our proposed δ^{EMD}. A key strategy here may be to look for domain-specific solutions.

Already however, the B^{EMD} as proposed offers many opportunities to incorporate domain expertise: in the choice of the candidate models, loss function, sensitivity parameter c, rejection threshold ϵ, and epistemic distributions used to validate c. These are clear interpretable choices that help express scientific intent. These choices, along with the assumptions underpinning the B^{EMD}, should also become *testable*: if they lead to selecting models that continue to predict well on new data, that in itself will provide for them a form of empirical validation.

5.4. Methods

5.4.1. Poisson noise model for black body radiation observations

In equation (5.38), we used a Poisson counting process to simulate the observation noise for recordings of a black body's radiance. This is a more realistic model than Gaussian noise for this system, while still being simple enough to serve our illustration.

The physical motivation is as follows. We assume that data are recorded with a spectrometer that physically separates photons of different wavelengths and measures their intensity with a CCD array. We further assume for simplicity that wavelengths are integrated in bins of equal width, such that the values of λ are sampled uniformly (the case with non-uniform bins is less concise but otherwise equivalent). We also assume that the device uses a fixed time window to integrate fluxes, such that what it detects are effective photon *counts*. The average number of counts is proportional to the radiance, but also to physical parameters of the sensor (including size, integration window and sensitivity) that we collect into the factor s; the units of s are $\mathrm{m}^2 \cdot \mathrm{nm} \cdot \text{photons} \cdot \mathrm{sr} \cdot \mathrm{kW}^{-1}$, such that $s\mathcal{B}_a(\lambda; T)$ is a number of photons. Since the photons are independent, the recorded number of photons will be random and follow a Poisson distribution. This leads to the following data-generating process $\mathcal{M}_{\mathrm{true}}$:

$$\mathcal{B} \mid \lambda, T \sim \frac{1}{s}\,\mathrm{Poisson}\big(s\,\mathcal{B}_{\mathrm{P}}(\lambda; T)\big) + \mathcal{B}_0\,, \qquad \text{(repeated from (5.39))}$$

$$\lambda \in \{\lambda_{\min}, \lambda_{\min} + \Delta\lambda, \lambda_{\min} + 2\Delta\lambda, \dots, \lambda_{\max}\}\,. \tag{5.49}$$

Here $\mathcal{B}_0$ captures the effect of dark currents and random photon losses on the radiance measurement. Recall that a random variable k following a distribution $\mathrm{Poisson}(\mu)$ has probability mass function $P(k) = \frac{\mu^k e^{-\mu}}{k!}$.

Note that we divide by s so that $\mathcal{B}$ also has dimensions of radiance and is comparable with the models $\mathcal{B}_{\mathrm{P}}$ and $\mathcal{B}_{\mathrm{RJ}}$ defined in Equations (5.36) and (5.37).

5.4.2. Neuron model

The neuron model used in our Results is the Hodgkin-Huxley-type model of the lobster pyloric rhythm studied by Prinz, Bucher, and Marder [21]. The specific implementation we used can be obtained from reference [183], along

with a complete description of all equations and parameters. We summarize the model below and refer the reader to that reference for more details.

For each cell k, the potential across a patch of area A and capacitance C evolves according to the net ionic current through the cell membrane:

$$\frac{C}{A}\frac{dV^k}{dt} = -\underbrace{\sum_{i\in\text{ion channels}} I_i^k}_{\substack{\text{ion diffusion}\\ \text{through membrane}}} - \underbrace{\sum_{l\in\text{neurons}} I_s^l}_{\substack{\text{chemical}\\ \text{synapses}}} - \underbrace{\sum_{l\in\text{neurons}} I_e^{kl}}_{\substack{\text{electrical}\\ \text{synapses}}} - I_{\text{ext}}^k\,, \tag{5.50}$$

where I_{ext} is an arbitrary external current, the I_i describe ion exchanges between a cell and its environment, and I_s and I_e describe ion exchanges between different cells. The I_{ext} current can be used to represent a current applied by the experimenter, or the inputs from other cells in the network. The voltage $\tilde{V}$ that an experimenter records would then be some corrupted version of V, subject to noise sources that depend on their experiment.

$$\tilde{V}(t) \sim \text{Experimental noise}\Big(V(t)\Big)\,. \tag{5.51}$$

We generate the external input I_{ext} as a Gaussian coloured noise with autocorrelation:

$$\Big\langle I_{\text{ext}}(t) I_{\text{ext}}(t')\Big\rangle = \sigma_i^2 e^{-(t-t')^2/2\tau^2}\,. \tag{5.52}$$

We do this using an implementation [184] of the sparse convolution algorithm [185]. We found that in addition to being more realistic, coloured noise also smears the model response in time and thus reduces degeneracies when comparing models.

Each cell in the model has eight currents through ion channels, indexed by i: one Na^+ current, two Ca^{2+} currents, four K^+ currents and one leak current. Each current is modelled as (square brackets indicate functional dependence)

$$\begin{aligned}
I_e^{kl} &= g_e\big(V^k - V^l\big)\,, & \tau_{m,i}\big[V^k\big]\frac{dm}{dt} &= m_{\infty,i}\big[V^k\big] - m_i^k\,,\\
I_i^k &= g_i^k \left(m_i^k\right)^{p_i} h_i \left(V^k - E_i\right), & \tau_{h,i}\big[V^k\big]\frac{dh}{dt} &= h_{\infty,i}\big[V^k\big] - h_i^k\,,\\
I_s^{kl} &= g_s^{kl}\, s^l \left(V^k - E_s^k\right), & \tau_s^l\big[V^l\big]\frac{ds}{dt} &= s_\infty^l\big[V^l\big] - s^l\,.
\end{aligned}$$

These equations are understood as describing currents through permeable channels with maximum conductivity g_i^k (for membrane currents within the same cells) or g_s^{kl} (for synaptic currents between different cells). Conductivities are dynamic: they are governed by the equations for the gating variables m, h and s given above. (Some channels do not have inactivating gates; for these, h_i is set to 1.) The fixed points m_∞, h_∞ and s_∞, as well as the time constants τ_m, τ_h and τ_s, are functions of the voltage; the precise shape of these functions is specific to each channel type and can be found in either references [21] or [183].

Following Prinz et al., we treat the functions m_∞, h_∞, s_∞, τ_m, τ_h and τ_s, as well as the electrical conductance g_e and the Nernst (E_i) and synaptic ($E_{s,k}$) reversal potentials, as known fixed quantities. Thus the only free parameters in this model are the maximum conductances g_i^k and g_s^{kl}: the former determine the type of each neuron, while the latter determine the circuit connectivity.

The pyloric circuit model studied by Prinz, Billimoria, and Marder [15] consists of three populations of neurons with eight different ion channels. Biophysically plausible values for the channel and connectivity parameters were determined through separate exhaustive parameter searches by Prinz et al. [15, 21], the results of which were reduced to sixteen qualitatively different parameter solutions: 5 *AB/PD* cells, 5 *LP* cells and 6 *PY* cells. Importantly, these parameter solutions are distinct: interpolating between them does not yield models that reproduce experimental recordings. For purposes of illustration we study the simple two-cell circuit shown in Fig. 5.2**a**, where an AB cell drives an LP cell. Moreover we assume the parameters of the AB cell to be known, such that we only need to compare model candidates for the LP cell. (These assumptions are not essential to applying our method, but they avoid us contending with model-specific considerations orthogonal to our exposition.)

Model symbol	Model components
$\mathcal{M}_{\text{true}}$	$I_{\text{ext}} \to$ AB3 $\to$ LP$\mathbb{1}$ $\to$ Gaussian noise $\to$ digitize
$\mathcal{M}_A$	$I_{\text{ext}} \to$ AB3 $\to$ LP$\mathbb{2}$ $\to$ Gaussian noise
$\mathcal{M}_B$	$I_{\text{ext}} \to$ AB3 $\to$ LP$\mathbb{3}$ $\to$ Gaussian noise
$\mathcal{M}_C$	$I_{\text{ext}} \to$ AB3 $\to$ LP$\mathbb{4}$ $\to$ Gaussian noise
$\mathcal{M}_D$	$I_{\text{ext}} \to$ AB3 $\to$ LP$\mathbb{5}$ $\to$ Gaussian noise

Table 5.2.: Neuron circuit model labels

The AB neuron is an autonomous pacemaker and serves to drive the circuit with realistic inputs; all of our examples use the same AB model (labelled 'AB/PD 3' in Table 2 of Prinz, Bucher, and Marder [21]), whose output is shown in Fig. 5.2**b**. Panels **c** and **d** show the corresponding response for each of the five LP models given in Table 2 of Prinz, Bucher, and Marder [21].

To generate our simulated observations, we use the output of `LP 1`, add Gaussian noise and then round the result (in millivolts) to the nearest 8-bit integer; in this way $\mathcal{M}_{\text{true}}$ includes both electrical and digitization noise. This leaves `LP 2` through `LP 5` to serve as candidate models; for these we assume only Gaussian noise, and we label them $\mathcal{M}_A$ to $\mathcal{M}_D$. We use Θ_a to denote the concatenation of all parameters for a given model a, which here consist of the vectors of conductance values g_i and g_s. Model definitions are summarized in Table 5.2 and Algorithm 4.

Algorithm 4 Neuron model

procedure GENERATE DATA($\mathcal{T}$)
  for $t \in \mathcal{T}$ **do**
    integrate Equation (5.50) to obtain $V^{\text{AB}}(t)$, $V^{\text{LP}}(t; \Theta_{\text{true}})$
    draw $\xi(t) \sim \mathcal{N}(0, \sigma_o)$
    evaluate $\tilde{V}^{\text{LP}}(t) = \texttt{digitize}_8\Big(V^{\text{LP}}(t) + \xi(t)\Big)$
  end for
  return $\big\{\tilde{V}^{\text{LP}}(t) : t \in \mathcal{T}\big\}$
end procedure

procedure SIMULATE CANDIDATE($\mathcal{T}$, $a \in \{A, B, C, D\}$)
  for $t \in \mathcal{T}$ **do**
    integrate Equation (5.50) to obtain $V^{\text{AB}}(t)$, $V^{\text{LP}}(t; \Theta_a)$
    draw $\xi(t) \sim \mathcal{N}(0, \sigma_o)$
    evaluate $\tilde{V}^{\text{LP}}(t) = V^{\text{LP}}(t; \Theta_a) + \xi(t)$.
  end for
  return $\big\{\tilde{V}^{\text{LP}}(t) : t \in \mathcal{T}\big\}$
end procedure

procedure $\texttt{digitize}_8$(x)
Simulates the data encoding of a digital sensor by converting x to an 8-bit integer
  clip: $x \leftarrow \texttt{max}\big(-128, \texttt{min}(x, 127)\big)$.
  return $\texttt{int}(x)$
end procedure

5.4.3. Loss function for the neuron model

To evaluate the risk of each candidate model, we use the log likelihood of the observations (equation (5.9)). This standard choice is convenient for exposition purposes: it is simple to explain and illustrates the generality of the method, since a likelihood function is available for any model in the form of equation (5.1).

However it is not a requirement to use the negative log likelihood as the loss, and in fact for time series models it can be disadvantageous. For example, the neuron models used in this work have sharp temporal responses (spikes), which makes the log likelihood sensitive to the timing of these spikes. In practice a less sensitive loss function may be preferable, although the best choice will depend on the application.

5.4.4. Construction of an R-distribution from an HB process $\mathfrak{Q}$

For each candidate model we use the hierarchical beta process $\mathfrak{Q}_a$ described below to generate on the order of $M_a \approx 100$ PPFs $\hat{q}_{a,1}, \dots, \hat{q}_{a,M_a}$; the exact number of PPFs is determined automatically, by increasing the number M_a until the relative standard error on $\frac{1}{M_a}\sum_{i=1}^{M_a} R[\hat{q}_{a,i}]$ is below 2^{-5} (six such PPFs are shown as grey traces in Fig. 5.5). Each curve is integrated to obtain a value for the risk (equation (5.20)), such that the R_a distribution can be represented by the set $\{R[\hat{q}_{a,i}]\}_{i=1}^{M_a}$. We then use a kernel density estimate to visualise these distributions in Fig. 5.3; specifically we use the `univariate_kde` function provided by Holoviews [186] with default parameters. The function automatically determines the bandwidth.

5.4.5. Calibration experiments

As described in our Results, the goal of calibration is twofold. First we want to correlate the value of $B^{\text{EMD}}_{AB;c}$ with the probability $B^{\text{epis}}_{AB;\Omega}$ that, across variations of the data-generating process $\mathcal{M}_{\text{true}}$, model A has lower risk than model B. We use an *epistemic distribution* Ω to describe those variations. Second, we want to ensure that a decision to reject either model based on B^{EMD}_{AB} is robust: it should hold for any reasonable epistemic distribution Ω of experimental conditions (and therefore hopefully also for *unanticipated* experimental variations).

The calibration procedure involves fixing two candidate models and an epistemic distribution Ω. We then sample multiple data-generating models $\mathcal{M}_{\text{true}}$ from Ω, generate a replicate dataset $\mathcal{D}^{\text{rep}}_{\text{test}}$ from each, and compute both B^{EMD} and B^{epis} on that replicate dataset.

This process can be repeated for as many epistemic distributions and as many different pairs of candidate models as desired, until we are sufficiently confident in the robustness of B^{EMD}. This means in particular that we cannot expect perfect correlation between B^{EMD} and B^{epis} across all models and conditions.

However we don't actually require perfect correlation because this is an *asymmetrical test*: a decision to reject *neither* model is largely preferred over a decision to reject the *wrong* one. Therefore what we especially want to ensure is that for any pair of models A, B, and any reasonable epistemic distribution Ω, we have

$$\left|B^{\text{EMD}}_{AB;c} - 0.5\right| \lesssim \left|B^{\text{epis}}_{AB;\Omega} - 0.5\right|. \tag{5.53}$$

This is given as either equation (5.34) or (5.35) in the Results.

Note that $B^{\text{epis}}_{AB;\Omega}$ is defined as the fraction of experiments (i.e. data-generating models $\mathcal{M}_{\text{true}}$) for which R_A is smaller than R_B. Therefore it cannot actually be calculated for a single experiment; only over sets of experiments. Since our goal is to identify a correlation between $B^{\text{epis}}_{AB;\Omega}$ and $B^{\text{EMD}}_{AB;c}$, we compute the former as a function of the latter:

$$\begin{aligned} B^{\text{epis}}_{AB;\Omega}\left(B^{\text{EMD}}_{AB;c}\right) &:= P\left(R_A < R_B \;\middle|\; \Omega, B^{\text{EMD}}_{AB;c}\right) \\ &= P_{\mathcal{M}_{\text{true}}\sim\Omega|B^{\text{EMD}}_{AB;c}}\left(R_A < R_B\right) \\ &= \mathbb{E}_{\mathcal{M}_{\text{true}}\sim\Omega|B^{\text{EMD}}_{AB;c}}\left[R_A < R_B\right]. \end{aligned} \tag{5.54}$$

(On the last line, [ ] is the Iverson bracket, which is 1 when the clause it contains is true, and 0 otherwise.) In other words, we partition the experiments in the domain of Ω into subsets having the same value $B^{\text{EMD}}_{AB;c}$, then compute $B^{\text{epis}}_{AB;\Omega}$ over each subset. In practice we do this by binning the experiments according to $B^{\text{EMD}}_{AB;c}$, as we explain below.

5.4.5.1. Calibration and computation of B^{epis} for the black body radiation models

For the black body models, we consider one epistemic distribution Ω with two sources of experimental variability: the true physical model ($\mathcal{B}_{\text{P}}$ or $\mathcal{B}_{\text{RJ}}$) and the bias $\mathcal{B}_0$. All other parameters are fixed to the standard values used throughout the text ($s = 10^5\,\text{m}^2 \cdot \text{nm} \cdot \text{photons} \cdot \text{sr} \cdot \text{kW}^{-1}$, $T = 4000\,\text{K}$).

It makes sense to use both candidate models to generate synthetic replicate datasets, since ostensibly both are plausible candidates for the true data generating process. This is also an effective way to ensure that calibration curves cover both high and low values of B^{EMD}.

To generate each replicate dataset $\mathcal{D}^{\text{rep}}_{\text{test}}$, we select

$$\mathcal{B}_{\text{phys}} \sim \text{Unif}\Big(\Big\{\mathcal{B}_{\text{P}}, \mathcal{B}_{\text{RJ}}\Big\}\Big) \tag{5.55}$$

$$\mathcal{B}_0 \sim \text{Unif}\Big(\big[-10^{-4}, 10^{-4}\big]\Big) \cdot \text{m}^2 \cdot \text{nm} \cdot \text{photons} \cdot \text{sr} \cdot \text{kW}^{-1} \tag{5.56}$$

We then generate 4096 data points for each dataset using the Poisson observation model of equation (5.38), substituting $\mathcal{B}_{\text{phys}}$ for $\mathcal{B}_{\text{P}}$.

As in the main text, both candidate models assume Gaussian noise, so for each replicate dataset and each candidate model we fit the standard deviation of that noise to the observation data by maximum likelihood.

We then compute $B^{\text{EMD}}_{\text{P,RJ};c} \in [0, 1]$ for each value of c being tested. We also compute a binary variable—represented again with the Iverson bracket—which is 1 if the true risk of $\mathcal{M}_{\text{P}}$ is lower risk than that of $\mathcal{M}_{\text{RJ}}$, and 0 otherwise:

$$\big[R_{\text{P}} < R_{\text{RJ}}\big] := \begin{cases} 1 & \text{if } R_{\text{P}} < R_{\text{RJ}}\,, \\ 0 & \text{otherwise.} \end{cases} \tag{5.57}$$

(In practice the true risks R_{P} and R_{RJ} are computed as the empirical risk with a very large number of data points.)

This produces a long list of $\Big(B^{\text{EMD}}_{\text{P,RJ};c}, \big[R_{\text{P}} < R_{\text{RJ}}\big]\Big)$ pairs, one for each dataset $\mathcal{D}^{\text{rep}}_{\text{test}}$. Formally, we can then compute $B^{\text{epis}}_{\text{P,RJ};\Omega}\big(B^{\text{EMD}}_{\text{P,RJ};c}\big)$ as per equation (5.54): by keeping only those pairs with matching $B^{\text{EMD}}_{\text{P,RJ};c}$ value and averaging $B^{\infty}_{\text{P,RJ}}$. In practice, all $B^{\text{EMD}}_{\text{P,RJ};c}$ values will be slightly different, so we bin neighbouring values together into a histogram. To make best use of our statistical power, we assign the same number of pairs to each bin, which means that bins have different widths; the procedure is as follows:

1. Sort all $\big(B^{\text{EMD}}_{\text{P,RJ};c}, [R_{\text{P}} < R_{\text{RJ}}]\big)$ pairs according to $B^{\text{EMD}}_{\text{P,RJ};c}$.
2. Bin the experiments by combining those with similar values of $B^{\text{EMD}}_{\text{P,RJ};c}$. Each bin should contain a similar number of experiments, so that they have similar statistical power.
3. Within each bin, average the values of $B^{\text{EMD}}_{\text{P,RJ};c}$ and $[R_{\text{P}} < R_{\text{RJ}}]$, to obtain a single pair $\big(\bar{B}^{\text{EMD}}_{\text{P,RJ};c}, \bar{B}^{\text{epis}}_{\text{P,RJ}}\big)$. These are the values we plot as calibration curves.

The unequal bin widths are clearly visible in Fig. 5.9, where points indicate the $\big(\bar{B}^{\text{EMD}}_{\text{P,RJ};c}, \bar{B}^{\text{epis}}_{\text{P,RJ}}\big)$ pair corresponding to each bin. Larger gaps between points are indicative of larger bins, which result from fewer experiments having B^{EMD} values close to $\bar{B}^{\text{EMD}}$.

Figure 5.9 is also a particularly clear illustration of the effect of the sensitivity factor c. When c is too small ($c \leq 2^{-9}$), R-distributions almost never overlap, and the $B^{\text{EMD}}_{\text{P,RJ};c}$ is mostly either 0 or 1: the criterion is *overconfident*, with many points located in the red regions. As c increases ($2^{-6} \leq c \leq 2^3$), $B^{\text{EMD}}_{\text{P,RJ};c}$ values become more evenly distributed over the entire $[0, 1]$ interval and either don't encroach at all into the overconfident regions, or do so sparingly. These curves show a nearly monotone relation between $B^{\text{EMD}}_{\text{P,RJ};c}$ and $B^{\text{epis}}_{\text{P,RJ};\Omega}$, indicating that the former is strongly predictive of the latter; the corresponding c values would likely be suitable for model comparison. In contrast, when c is too large ($c \geq 2^6$), B^{EMD} becomes decorrelated from B^{epis}: all points are near $B^{\text{epis}} \approx 0.5$, indicating that the $B^{\text{EMD}}_{\text{P,RJ};c}$ for such large values of c would not be a useful comparison criterion.

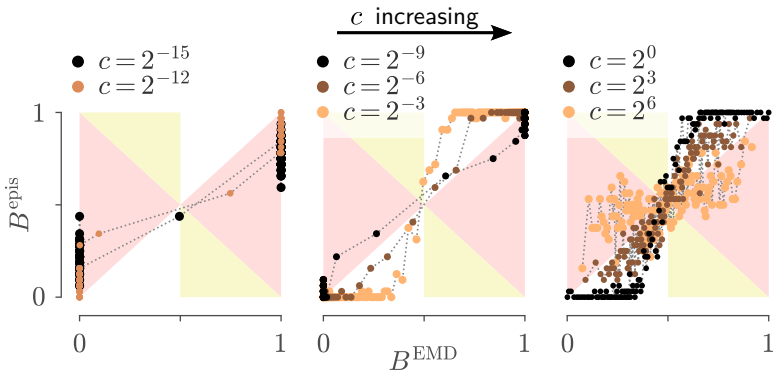


Figure 5.9.: Calibration plots for the $B^{\text{EMD}}_{\text{P,RJ}}$ criterion comparing Planck and Rayleigh-Jeans models. The value of c increases from left to right and dark to bright. Each curve consists of 4096 experiments aggregated into 128 bins. Wide gaps between points indicate that few experiments produced similar values of B^{EMD}; this is especially salient in the left panel, where almost all experiments have either $B^{\text{EMD}} = 0$ or $B^{\text{EMD}} = 1$.

5.4.5.2. Calibration for the neural response models

For the neuron model, we consider four sources of experimental variability: variations in the distribution used to model observation noise ξ, variations in the strength (σ_o) of observation noise, as well as variations in the strength (σ_i) and correlation time (τ) of the external input I_{ext}. Each epistemic distribution Ω is therefore described by four distributions over hyperparameters:

Observation noise model One of *Gaussian* or *Cauchy*. The distributions are centered and σ_o is drawn from the distribution defined below. Note that this is the model used to *generate* a dataset $\mathcal{D}^{\text{rep}}_{\text{test}}$. The candidate models always evaluate their loss assuming a Gaussian observation model.

$$\underset{\text{Gaussian}}{p(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\xi^2}{2\sigma_o^2}\right)} \qquad \underset{\text{Cauchy}}{p(\xi) = \frac{2}{\pi\sigma\left[1+\left(\frac{\xi^2}{\sigma_o/2}\right)\right]}}$$

Observation noise strength σ_o

Low noise:	$\log \sigma_o \sim \mathcal{N}(0.0\,\text{mV}, (0.5\,\text{mV})^2)$
High noise:	$\log \sigma_o \sim \mathcal{N}(1.0\,\text{mV}, (0.5\,\text{mV})^2)$

External input strength σ_i The parameter σ_i sets the strength of the input noise such that $\langle I^2_{\text{ext}} \rangle = \sigma_i^2$.

Weak input:	$\log \sigma_i \sim \mathcal{N}(-15.0\,\text{mV}, (0.5\,\text{mV})^2)$
Strong input:	$\log \sigma_i \sim \mathcal{N}(-10.0\,\text{mV}, (0.5\,\text{mV})^2)$

External input correlation time τ The parameter τ sets the correlation time of the input noise such that $\langle I_{\text{ext}}(t) I_{\text{ext}}(t+s) \rangle = \sigma_i^2 e^{-s^2/2\tau^2}$.

Short correlation:	$\log_{10} \tau \sim \text{Unif}([0.1\,\text{ms}, 0.2\,\text{ms}])$
Long correlation:	$\log_{10} \tau \sim \text{Unif}([1.0\,\text{ms}, 2.0\,\text{ms}])$

We thus defined 2 statistical distributions for ξ, 2 distributions for σ_o, 2 distributions for τ, and 2 distributions for σ_i. Combined with three model pairs (see Fig. 5.6), this makes a total of $48 = 3 \times 2 \times 2 \times 2 \times 2$ possible epistemic distributions Ω, each of which can be identified by a tuple such as ($\mathcal{M}_A$ vs $\mathcal{M}_B$, `Cauchy`, `Low noise`, `Strong input`, `Long correlation`). Calibration results for each of these conditions are given in Supplementary Fig. 5.10.

During calibration against an epistemic distribution Ω, drawing a dataset $\mathcal{D}^{\text{rep}}_{\text{test}}$ happens in two steps. First, we randomly draw a vector ω of epistemic parameters (i.e. hyperparameters) from Ω; for example $\omega = (\underbrace{-0.27\text{mV}}_{\sigma_o}, \underbrace{47\text{ms}}_{\tau}, \underbrace{0.003\text{mV}}_{\sigma_i})$. Second, we use those parameters to generate the dataset $\mathcal{D}^{\text{rep}}_{\text{test}}$, composed in this case of data points $(t, \tilde{V}^{\text{LP}}(t))$. We can then evaluate the loss Q (given by equation (5.9)) on those data points. Note that the loss does not depend on the epistemic parameters ω directly, but in general will involve parameters that are fitted to the simulated data. In short, the vector ω describes the parameters of a simulated experiment, while $\mathcal{D}^{\text{rep}}_{\text{test}}$ describes a particular outcome of that experiment. In theory we could generate multiple datasets with the same vector ω, but in practice it is more statistically efficient to draw a new experiment for each new dataset.

When choosing epistemic distributions, it is worth remembering that the goal of calibration is to empirically approximate a probability over experimental conditions. Thus choosing a distribution that can generate a large number of conditions—ideally an infinite number—will lead to better estimates. Here the use of continuous distributions for σ_o, σ_i and τ, and the fact that I_{ext} is a continuous process, helps us achieve this goal. Also important is to generate data where the model selection problem is ambiguous, so as to resolve calibration curves around $B^{\text{EMD}} = 0.5$.

This calibration is an imperfect procedure, and how well it works depends on the quality of the candidate models and the choice of loss function. Here for example, the choice of a pointwise loss makes it sensitive to the timing of spikes; this tends to favour models that produce fewer spikes, since the penalty on a mistimed spike is high. This is why in Fig. 5.6, in the $\mathcal{M}_A$ vs $\mathcal{M}_D$ comparison, we see a floor on the values of B^{epis}_{AD}. In short, some models have consistently lower loss even on random data, and so their risk—which is the expectation of their loss—is a priori lower. The bias we see in the $\mathcal{M}_A$ vs $\mathcal{M}_B$ comparison is likely due to a similar effect. (In this case because $\mathcal{M}_B$ is slightly better than $\mathcal{M}_A$ on average.)

5.4.6. The hierarchical beta process

In this work we identify the epistemic uncertainty of a model $\mathcal{M}_A$ with the variability of a stochastic process $\mathfrak{Q}_A$: realizations of $\mathfrak{Q}_A$ approximate the PPF of the model loss. In our Results we listed desiderata that $\mathfrak{Q}_A$ should satisfy and proposed that it be described as a hierarchical beta (HB) process. However we deferred providing a precise definition for $\mathfrak{Q}_A$; we do this now in the form of 5. The rest of this section explains the theoretical justifications for each step of this generative algorithm for $\mathfrak{Q}_A$.

5.4.6.1. Relevant concepts of Wiener processes

Before introducing the HB process, let us first review a few key properties that stochastic processes must satisfy and that are covered in most standard introductions [120, 187, 188]. We use for this the well-known Wiener process, and also introdue notation that will become useful when we define the HB process. Since our goal is to define a process for PPFs, we use Φ to denote the independent "domain" variable and restrict ourselves to 1-d processes for which $\Phi \in [0, 1]$.

For the Wiener process $\mathcal{W}$, each realization is a continuous function $W : [0, 1] \to \mathbb{R}$. One way to approximate a realization of $\mathcal{W}$ is to first partition the interval into subintervals $[0, \Phi_1)$, $[\Phi_1, \Phi_2)$, …, $[\Phi_n, 1)$ with $0 < \Phi_1 < \Phi_2 < \cdots < \Phi_n < 1$; for simplicity we will only consider equal-sized subintervals, so that $\Phi_k = k\,\Delta\Phi$ for some $\Delta\Phi \in \mathbb{R}$. We then generate a sequence of independent random increments (one for each subinterval) $\{\Delta W_{\Delta\Phi}(0), \Delta W_{\Delta\Phi}(\Delta\Phi), \Delta W_{\Delta\Phi}(2\Delta\Phi), \dots, \Delta W_{\Delta\Phi}(1 - \Delta\Phi)\}$ and define the corresponding realization as

$$W(k\,\Delta\Phi) = W(0) + \sum_{l=0}^{k-1} \Delta W_{\Delta\Phi}(l\,\Delta\Phi) \tag{5.58}$$

(Within each interval the function may be linearly interpolated, so that W is continuous.)

A *refinement* of a partition is obtained by taking each subinterval and further dividing it into smaller subintervals. For instance we can refine the unit interval $[0, 1)$ into a set of two subintervals, $[0, 2^{-1})$ and $[2^{-1}, 2^{-0})$. Let us denote

Algorithm 5 Hierarchical beta process

Given

- q_A^*, c and δ_A^{EMD}, ▷ computed from data
- $N \in \mathbb{N}^+$, ▷ number of refinements
- $p(\hat{q}(0), \hat{q}(1))$, ▷ 2-d distribution over end points

generate a discretized realization $\hat{q}_A$.

Procedure:

▷ Initialize the procedure by drawing end point
repeat
  draw $\big(\hat{q}(0), \hat{q}(1)\big) \sim p\big(\hat{q}(0), \hat{q}(1)\big)$
until $\hat{q}(0) < \hat{q}(1)$ ▷ PPFs must be increasing
▷ Successively refine the interval
for $n \in 1, 2, \dots, N$ **do** ▷ refinement levels
  for $\Phi \in \{k \cdot 2^{-n+1}\}_{k=0}^{2^{n-1}-1}$ **do** ▷ intermediate incrs.
    compute r, v according to equations (5.73)
    solve equations (5.75) to obtain α and β
    draw $x_1 \sim \mathrm{B}(\alpha, \beta)$
    $\hat{q}(\Phi + 2^{-n}) \leftarrow \hat{q}(\Phi) + \Delta\hat{q}_{\Delta\Phi}(\Phi) \cdot x_1$
  end for
end for
return $\big(\hat{q}(\Phi) \,:\, \Phi \in \{\mathcal{I}_\Phi\}^{(N)}\big)$

The quantities r and v computed on Algorithm 5 conceptually represent the **r**atio between two sucessive increments and the **v**ariance of those increments.

these partitions $\{\mathcal{I}_\Phi\}^{(0)}$ and $\{\mathcal{I}_\Phi\}^{(1)}$ respectively. Repeating the process on each subinterval yields a sequence of ever finer refinements:

$$\begin{aligned}
\{\mathcal{I}_\Phi\}^{(0)} &\coloneqq \Big\{\big[\,0,\ 2^{-0}\,\big)\Big\} \\
\{\mathcal{I}_\Phi\}^{(1)} &\coloneqq \Big\{\big[\,0,\ 2^{-1}\,\big),\ \big[\,2^{-1},\ 2^{-0}\,\big)\Big\} \\
\{\mathcal{I}_\Phi\}^{(2)} &\coloneqq \Big\{\big[\,0,\ 2^{-2}\,\big),\ \big[\,2^{-2},\ 2^{-1}\,\big),\ \big[\,2^{-1},\ 3\cdot 2^{-2}\,\big),\ \big[\,3\cdot 2^{-2},\ 2^{-0}\,\big)\Big\} \\
&\ \ \vdots \\
\{\mathcal{I}_\Phi\}^{(n)} &\coloneqq \Big\{\big[\,k\cdot 2^{-n},\ (k{+}1)\cdot 2^{-n}\,\big)\Big\}_{k=0}^{2^n-1}.
\end{aligned} \tag{5.59}$$

With these definitions, for any $m \geq n$, $\{\mathcal{I}_\Phi\}^{(m)}$ is a refinement of $\{\mathcal{I}_\Phi\}^{(n)}$.

Later we will need to refer to the vector of new end points introduced at the n-th refinemement step. These are exactly the odd multiples of 2^{-n} between 0 and 1, which we denote $\{\Phi\}^{(n)}$:

$$\{\Phi\}^{(n)} \coloneqq \Big(2^{-n},\ 3\cdot 2^{-n},\ \dots\ ,(2^n{-}3)\cdot 2^{-n},\ (2^n{-}1)\cdot 2^{-n}\Big). \tag{5.60}$$

Any random process must be *self-consistent* [166]: for small enough $\Delta\Phi$, the probability distribution at a point Φ must not depend on the level of refinement. For example, the Wiener process is defined such that the increments $\Delta W_{\Delta\Phi} \sim \mathcal{N}(0, \Delta\Phi)$ are independent; therefore

$$\begin{aligned}
&W(\Phi{+}2\Delta\Phi) \\
&= W(\Phi) + \Delta W_{2\Delta\Phi}(\Phi) \ \ = W(\Phi) + \Delta W_{\Delta\Phi}(\Phi) \ + \Delta W_{\Delta\Phi}(\Phi{+}\Delta\Phi) \\
&= W(\Phi) + \mathcal{N}\big(0, 2\Delta\Phi\big) = W(\Phi) + \mathcal{N}\big(0, \Delta\Phi\big) + \mathcal{N}\big(0, \Delta\Phi\big).
\end{aligned} \tag{5.61}$$

It turns out that the combination of the Markovian and self-consistent properties set quite strong requirements on the stochastic increments, since they impose the square root scaling of the Wiener increment: $\mathcal{O}(\Delta W_{\Delta\Phi}) = \mathcal{O}(\sqrt{\Delta\Phi})$. (§II.C of reference [166].)

While the Wiener process underlies much of stochastic theory, it is not suitable for defining a process $\mathfrak{Q}$ over PPFs. Indeed, it is not monotone by design, which violates one of our desiderata. Moreover, it has a built-in directionality in the form of accumulated increments. A clear symptom of this is that as Φ increases, the variance of $W(\Phi)$ also increases. (This follows immediately from equation (5.58) and the independence of increments.) Directionality makes sense if we think of W as modelling the diffusion of particles in space or time, but empirical PPFs are obtained by first sorting data samples according to their loss (see the definition of δ^{EMD} in the Results). Since samples of the loss arrive in no particular order, a process that samples PPFs should likewise have no intrinsic directionality in Φ.

5.4.6.2. A hierarchical beta distribution is monotone, non-accumulating and self-consistent

Constructing a stochastic process $\mathfrak{Q}$ that is *monotone* is relatively simple: one only needs to ensure that the random increments $\Delta\hat{q}_{\Delta\Phi}(\Phi)$ are non-negative.

Ensuring that those increments are *non-accumulating* requires more care, because that requirement invalidates most common definitions of stochastic processes. As described in our Results, we achieve this by defining $\mathfrak{Q}$ as a sequence of refinements, starting from a single increment for the entire interval, then doubling the number of increments (and halving their width) at each refinement step. In the rest of this subsection we give an explicit construction of this process and show that it is also *self-consistent*. (Altough in this work we consider only pairs of increments sampled from a beta distribution, in general one could consider other compositional distributions. Higher-dimensional distributions may allow to sample all increments simultaneously, if one can determine the conditions that ensure self-consistency.)

For an interval $\mathcal{I} = [\Phi, \Phi + \Delta\Phi)$, we suppose that the points $\hat{q}_A(\Phi)$ and $\hat{q}_A(\Phi + \Delta\Phi)$ are given. We define

$$\Delta\hat{q}(\mathcal{I}) \;=\; \Delta\hat{q}_{\Delta\Phi}(\Phi) \;\coloneqq\; \hat{q}_A(\Phi + \Delta\Phi) - \hat{q}_A(\Phi)\,, \tag{5.62}$$

then we draw a subincrement $\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi)$, associated to the subinterval $[\Phi, \Phi + \Delta\Phi)$, from a scaled beta distribution:

$$\frac{1}{\Delta\hat{q}_{\Delta\Phi}(\Phi)}\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi) \coloneqq x_1 \sim \mathrm{B}(\alpha, \beta)\,. \tag{5.63}$$

(Refer to the subsection below for the definition of $\mathrm{B}(\alpha, \beta)$.) The scaling is chosen so that

$$0 \leq \underbrace{\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi)}_{= x_1 \Delta\hat{q}_{\Delta\Phi}(\Phi)} \leq \Delta\hat{q}_{\Delta\Phi}(\Phi)\,. \tag{5.64}$$

The value of $\Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi)$ then determines the intermediate point:

$$\hat{q}_A\big(\Phi + \tfrac{\Delta\Phi}{2}\big) \leftarrow \hat{q}_A(\Phi) + \Delta\hat{q}_{\frac{\Delta\Phi}{2}}(\Phi)\,. \tag{5.65}$$

If desired, the complementary increment $\Delta\hat{q}_{\frac{\Delta\Phi}{2}}\big(\Phi + \frac{\Delta\Phi}{2}\big)$ can be obtained as

$$\Delta\hat{q}_{\frac{\Delta\Phi}{2}}\big(\Phi + \tfrac{\Delta\Phi}{2}\big) = (1 - x_1)\Delta\hat{q}_{\Delta\Phi}(\Phi)\,. \tag{5.66}$$

Generalizing the notation to the entire $[0, 1)$ interval, we start from a sequence of increments associated to subintervals at refinement step n (recall 5.59):

$$\begin{aligned}\{\Delta\hat{q}_{2^{-n}}\} &\coloneqq \big\{\Delta\hat{q}(\mathcal{I}) \quad : \quad \mathcal{I} \in \{\mathcal{I}_\Phi\}^{(n)}\big\} \\ &\overset{\text{def}}{=} \big\{\Delta\hat{q}_{2^{-n}}(\Phi) \,:\big[\Phi, \Phi + 2^{-n}\big) \in \{\mathcal{I}_\Phi\}^{(n)}\big\}.\end{aligned}$$

Applying the procedure just described, for each subinterval we draw x_1 and split the corresponding increment $\Delta\hat{q}_{2^{-n}}(\Phi) \in \{\Delta\hat{q}_{2^{-n}}\}$ into a pair $\big(\Delta\hat{q}_{2^{-n-1}}(\Phi), \Delta\hat{q}_{2^{-n-1}}(\Phi + 2^{-n-1})\big)$ of subincrements such that

$$\Delta\hat{q}_{2^{-n}}(\Phi) = \Delta\hat{q}_{2^{-n-1}}(\Phi) + \Delta\hat{q}_{2^{-n-1}}(\Phi + 2^{-n-1})\,. \tag{5.67}$$

The union of subincrements is then the next refinement step:

$$\big\{\Delta\hat{q}(\mathcal{I}) \,:\, \mathcal{I} \in \{\mathcal{I}_\Phi\}^{(n+1)}\big\} = \{\Delta\hat{q}_{2^{-n-1}}\}\,. \tag{5.68}$$

After n refinement steps, we thus obtain a function $\hat{q}(\Phi)$ defined at discrete points:

$$\hat{q}^{(n)}(k \cdot 2^{-n}) \coloneqq \hat{q}(0) + \sum_{l<k} \Delta\hat{q}_{2^{-n}}(l \cdot 2^{-n})\,, \tag{5.69}$$

which we extend to the entire interval $[0, 1)$ by linear interpolation; see Fig. 5.5**d** for an illustration. In practice we found that computations (specifically the risk computed by integrating $\hat{q}^{(n)}(\Phi)$) converge after about eight refinement steps.

This procedure has the important property that once a point is sampled, it does not change on further refinements:

$$\hat{q}^{(n)}(k \cdot 2^{-n}) = \hat{q}^{(m)}(k \cdot 2^{-n})\,, \quad \forall m \geq n\,, \tag{5.70}$$

which follows from equation (5.67). Recall now that, as stated below Equation (5.60), a process is self-consistent if "for small enough $\Delta\Phi$, the probability distribution at a point Φ [does] not depend on the level of refinement". Since equation (5.70) clearly satisfies that requirement, we see that the process obtained after infinitely many refinement steps is indeed self-consistent. We thus define the *hierarchical beta (HB) process* $\mathfrak{Q}_A$ as

$$p(\hat{q} \mid \mathfrak{Q}_A) \coloneqq p\Big(\hat{q} = \lim_{n\to\infty} \hat{q}^{(n)} \;\Big|\; c, q^*_A, \delta^{\text{EMD}}_A\Big)\,. \tag{5.71}$$

To complete the definition of $\mathfrak{Q}_A$, we need to specify how we choose the initial end points $\hat{q}_A(0)$ and $\hat{q}_A(1)$. In our code implementation, they are drawn from normal distributions $\mathcal{N}\big(q^*(\Phi), \sqrt{c}\,\delta^{\text{EMD}}_A(\Phi)^2\big)$ with $\Phi \in \{0, 1\}$, where again c is determined via our proposed calibration procedure; this is simple and convenient, but otherwise arbitrary. We also need to explain how we choose the beta parameters α and β, which is the topic of the next subsection.

5.4.6.3. Choosing beta distribution parameters

All HB processes are monotone, continuous and self-consistent, but within this class there is still a lot of flexibility: since α and β are chosen independently for each subinterval, we can mold $\mathfrak{Q}_A$ into a wide variety of statistical shapes. We use this flexibility to satisfy the two remaining desiderata: a) that $\hat{q}_A(\Phi)$ realizations track $q^*_A(\Phi)$ over $\Phi \in [0, 1]$; and b) that the variability of $\hat{q}_A(\Phi)$ be proportional to $\delta^{\text{EMD}}_A(\Phi)$. It is the goal of this subsection to give a precise mathematical meaning to those requirements.

Let $x_1 \sim \text{B}(\alpha, \beta)$ and $x_2 = 1 - x_1$. (The density function of a beta distribution is given in 5.24.) The mean and variance

of x_1 are

$$\mathbb{E}[x_1] = \frac{\alpha}{\alpha + \beta}\,, \tag{5.72a}$$

$$\mathbb{V}[x_1] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}\,. \tag{5.72b}$$

For a given Φ, it may seem natural to select α and β by matching $\mathbb{E}[x_1]$ to $q_A^*(\Phi)$ and $\mathbb{V}[x_1]$ to $c\cdot\big(\delta_A^{\mathrm{EMD}}(\Phi)\big)^2$. However both equations are tightly coupled, and we found that numerical solutions were unstable and unsatisfactory; in particular, it is not possible to make the variance large when $\mathbb{E}[x_1]$ approaches either 0 or 1 (otherwise the distribution of x_1 would exceed $[0,1]$).

Much more practical is to consider moments with respect to the Aitchison measure; as mentioned in the Results, the Aitchison measure first maps the bounded interval $[0,1]$ to the unbounded space $\mathbb{R}$ with a logistic transformation, then computes moments in the unbounded space. The first two such moments are called the *centre* and the *metric variance* [167, 168]; for the beta distribution, they are given by (reproduced from 5.25)

$$\mathbb{E}_a[(x_1,x_2)] = \frac{1}{e^{\psi(\alpha)}+e^{\psi(\beta)}}\big(e^{\psi(\alpha)}, e^{\psi(\beta)}\big)\,, \tag{5.25a}$$

$$\mathrm{Mvar}[(x_1,x_2)] = \frac{1}{2}\big(\psi_1(\alpha)+\psi_1(\beta)\big)\,, \tag{5.25b}$$

where ψ and ψ_1 are the digamma and trigamma functions respectively. The centre and metric variance are known to be more natural statistics for compositional distributions, and this is what we found in practice. Therefore we will relate q_A^* to the centre, and $\sqrt{c}\,\delta_A^{\mathrm{EMD}}$ to the square root of the metric variance.

To be precise, suppose that we have already selected a set of increments $\{\Delta\hat{q}_{2^{-n}}\}$ over the domain $\{k\cdot 2^{-n}\}_{k=0}^{n-1}$, and wish to produce the refinement $\{\Delta\hat{q}_{2^{-n-1}}\}$. For each Φ in $\{k\cdot 2^{-n}\}_{k=0}^{n-1}$ we define

$$r \coloneqq \frac{q_A^*(\Phi+2^{-n-1}) - q_A^*(\Phi)}{q_A^*(\Phi+2^{-n}) - q_A^*(\Phi+2^{-n-1})}\,, \tag{5.73a}$$

$$v \coloneqq 2\,c\left(\delta_A^{\mathrm{EMD}}\big(\Phi+2^{-n-1}\big)\right)^2. \tag{5.73b}$$

The value r is the ratio of subincrements of $\Delta q_{2^{-n}}^*(\Phi)$. Since we want $\hat{q}_A$ to track q_A^*, it makes sense to expect r to also approximate the ratio of subincrements of $\Delta\hat{q}_{2^{-n}}(\Phi)$. Identifying

$$r \leftrightarrow \frac{\mathbb{E}_a[x_1]}{\mathbb{E}_a[x_2]} \quad\text{and}\quad v \leftrightarrow 2\cdot\mathrm{Mvar}[(x_1,x_2)]\,, \tag{5.74}$$

and substituting into equations (5.25) then leads to the following system of equations:

$$\psi(\alpha) - \psi(\beta) = \ln r \tag{5.75a}$$

$$\ln\big[\psi_1(\alpha)+\psi_1(\beta)\big] = \ln v\,, \tag{5.75b}$$

which we can solve to yield the desired parameters α and β for each subinterval in $\{\mathcal{I}_\Phi\}^{(n)}$.

In summary, although the justification is somewhat technical, the actual procedure for obtaining α and β is quite simple: first compute r and v following equations (5.73), then solve equations (5.75).

Data availability

All figure data can be regenerated from code (see Code availability). Results of especially long computations are included as precomputed data files alongside the accompanying computational notebooks [189].

Code availability

All source code used to produce the figures in this paper is available as a collection of Jupyter notebooks [189]. Additional Python code for simulating the neuron circuit model [183] and generating colored noise [184] is also available.

We also provide the Python package `emdcmp` [158], available on the Python Packaging Index (PyPI), which automates many of the computation steps in our approach: given a set of observed data $\mathcal{D}_{\text{test}}$, a generative model $\mathcal{M}_A$, a loss function Q and sensitivity parameter $c \in \mathbb{R}$, `emdcmp` will automatically compute $\tilde{q}_A$, q_A^* and δ_A^{EMD}, then draw samples from the corresponding R_A-distribution. In other words, through `emdcmp` we provide a standard implementation for all the steps represented by downward facing arrows on the right of Fig. 5.1.

The `emdcmp` package also provides utilities to help execute calibration experiments.

Acknowledgements

We thank Anno Kürth, Aitor Morales-Gregorio, Günther Palm, Moritz Helias, Jan Bölts, Abel Jansma, Thomas Nowotny and Manfred Opper for helpful comments and discussions.

This work was partly supported by the German Federal Ministry for Education and Research (BMBF grant no. 01IS19077B; AR), the Natural Sciences and Engineering Research Council of Canada (NSERC; AL), the government of Ontario (OGS; AR), and the European Union (ERC grant "HIGH-HOPeS", no. 101039827; AR).

Supplementary

5.A. Supplementary Methods

5.A.1. Transitivity of B^{EMD} comparisons

Given three independent continuous random variables R_A, R_B and R_C, define the probabilities

$$\underbrace{P(R_A < R_B)}_{=:B_{AB}} \qquad \underbrace{P(R_B < R_C)}_{=:B_{BC}} \qquad \underbrace{P(R_C < R_A)}_{=:B_{CA}} .$$

For some threshold ϵ, we would like these to satisfy a transitivity relation of the form

$$\left.\begin{aligned} B_{AB} > \epsilon \\ B_{BC} > \epsilon \end{aligned}\right\} \Rightarrow B_{CA} < \epsilon , \tag{5.76}$$

as this would reduce the required number of pairwise comparisons between models (see section Model discrepancy as a baseline for non-stationary replications, Table 5.1 and equation (5.17) in the main text).

It is known that equation (5.76) does not hold for $\epsilon = \frac{1}{2}$; classic counterexamples in this case are non-transitive dice [190]. However, the set of probabilities $\mathcal{S} := \{B_{AB}, B_{BC}, B_{CA}\}$ *does* satisfy a property known as *dice-transitivity* [164], from which one can derive that equation (5.76) holds when $\epsilon = \varphi^{-1}$, where φ is the golden ratio. This result appears as a comment below Theorem 3 in Baets and Meyer [165], but to our knowledge has otherwise remained unknown. We provide a short self-contained derivation below, for the convenience of the reader.

The definition of dice-transitivity is obtained by substituting equation (9) of De Schuymer, De Meyer, and De Baets [164] into equation (6) of the same reference. For our purposes we are interested in the resulting upper bound

$$\alpha - 1 \leq -\beta\gamma , \tag{5.77}$$

where α, β, and γ are respectively the lowest, middle and highest value of $\mathcal{S}$. In other words, $\{\alpha, \beta, \gamma\} = \mathcal{S}$ and

$$\alpha \leq \beta \leq \gamma . \tag{5.78}$$

Suppose that, as given in equation (5.17), we have

$$B_{AB} > \varphi^{-1} \quad \text{and} \quad B_{BC} > \varphi^{-1} . \tag{5.79}$$

We wish to use equation (5.77) to establish an upper bound on B_{CA}. We do not know a priori how the probabilities are ordered, so we consider the six possible cases:

B_{AB}	B_{BC}	B_{CA}
α	β	γ
β	α	γ
α	γ	β
γ	α	β
β	γ	α
γ	β	α

(5.80)

Cases $\alpha\beta\gamma$ and $\beta\alpha\gamma$. The assumptions of equation (5.79) translate to $\alpha > \varphi^{-1}$ and $\beta > \varphi^{-1}$. We seek a bound on γ. Rearranging equation (5.77), we then have

$$\gamma \leq \frac{1-\alpha}{\beta} < \frac{1-\varphi^{-1}}{\varphi^{-1}} = \frac{-1+\sqrt{5}}{2} = \varphi^{-1}\,, \tag{5.81}$$

which contradicts equation (5.78).

Cases $\alpha\gamma\beta$ and $\gamma\alpha\beta$. The argument is exactly analogous, except that we seek a bound on β. We get

$$\beta \leq \frac{1-\alpha}{\gamma} < \frac{1-\varphi^{-1}}{\varphi^{-1}} = \varphi^{-1}\,, \tag{5.82}$$

which again contradicts equation (5.78).

Therefore the only possible cases are $\beta\gamma\alpha$ and $\gamma\beta\alpha$, which means that B_{CA} must be the smallest of the three probabilities. These two final cases provide the upper bound on B_{CA}:

Cases $\beta\gamma\alpha$ and $\gamma\beta\alpha$. We seek a bound on α. Rearranging equation (5.77) one more time yields

$$\alpha \leq 1 - \beta\gamma < 1 - \varphi^{-2} = \varphi^{-1}\,. \tag{5.83}$$

Thus equation (5.76) holds for $\epsilon = \varphi^{-1}$. More generally, we have $\alpha = B_{CA}$ whenever both B_{AB} and B_{BC} are greater than φ^{-1}. In this case equation (5.83) implies

$$B_{AC} = 1 - B_{CA} > 1 - (1 - B_{AB}B_{BC}) = B_{AB}B_{BC}\,, \tag{5.84}$$

and therefore

$$\left.\begin{aligned} B_{AB} &> \varphi^{-1} \\ B_{BC} &> \varphi^{-1} \end{aligned}\right\} \Rightarrow\ B_{AC} > B_{AB}B_{BC}\,. \tag{5.85}$$

Equation (5.17) given in the main text is a special case of equation (5.85).

5.B. Supplementary Results

5.B.1. Additional calibration curves for the neuron model

In Calibrating and validating the B^{EMD}, we considered only the effect of varying the external input strength (σ_i) on the calibration curves for the Prinz model. Figure 5.10 is an extended version of Fig. 5.6, where all 48 proposed epistemic distributions are plotted.

5.B.2. Sensitivity factor c can shift R-distributions

The primary effect of the sensitivity parameter c in Equations (5.16) and (5.27) is to increase the spread of R-distributions, and thus their overlap. However, in contrast to the case where we simply add additive noise (e.g. the noise η in equation (5.101)), the parameter c can also affect the shape of R-distributions, and in particular can change the relative distance between their centres. This is illustrated in Supplementary Fig. 5.11.

This phenomenon is one reason why the calibration curves actually change shape when we change c (c.f. Supplementary Fig. 5.10 and 5.13), and therefore that it is worthwhile to pin down an appropriate value for c. It is best understood as an effect of the additional constraints on $\mathfrak{Q}$: that realisations be monotone, integrable, and

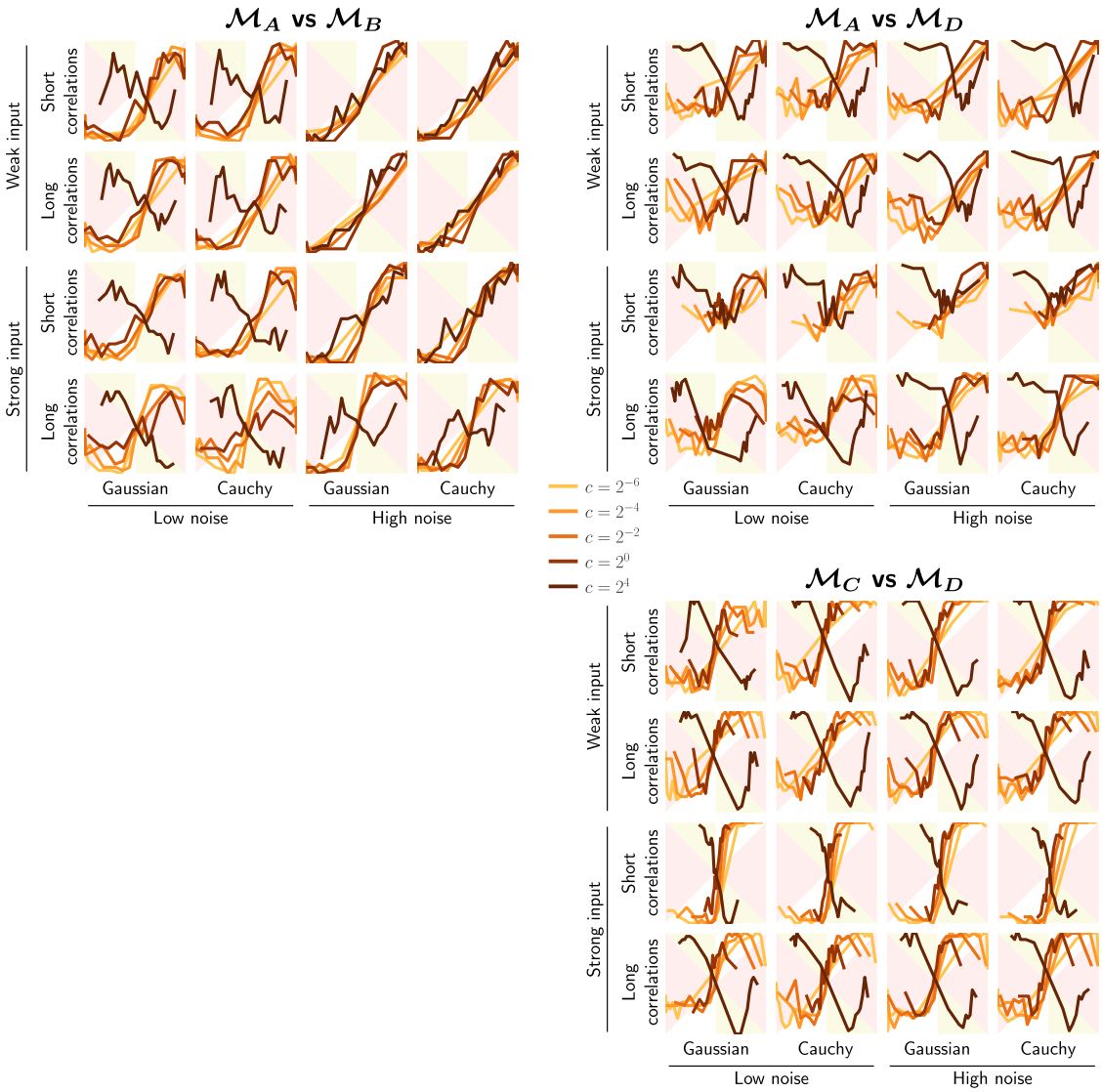


Figure 5.10.: Additional calibration curves for the neuron model, computed following the procedure described in section Calibrating and validating the B^{EMD} of the main text. Each curve summarizes 512 simulated experiments with datasets of size 4000. The regions depicted in red and yellow are those where equation (5.34) is violated.

non-accumulating. These can change the shape of the $\mathfrak{Q}$ distribution, beyond the simple linear spread around q^* defined by 5.27.

This underscores also the importance of considering those constraints to correctly sample the space of experimental variations.

5.B.3. Comparison of selection criteria with inconclusive data

A key motivation underlying the B^{EMD} was to define a selection criterion that provides reliable estimates of its uncertainty, in particular for large test datasets (thousands of samples or more): these are commonly produced by experiments and yet produce pathologies with many of the common statistical methods of model selection. In short, a criterion can both over and underestimate the strength of the evidence in favour of one model.

We illustrate this phenomenon in Supplementary Table 5.3, with a set of comparisons meant to complement those of Fig. 5.8 in the main text and which explore the effect of misspecification in greater detail. We use 18

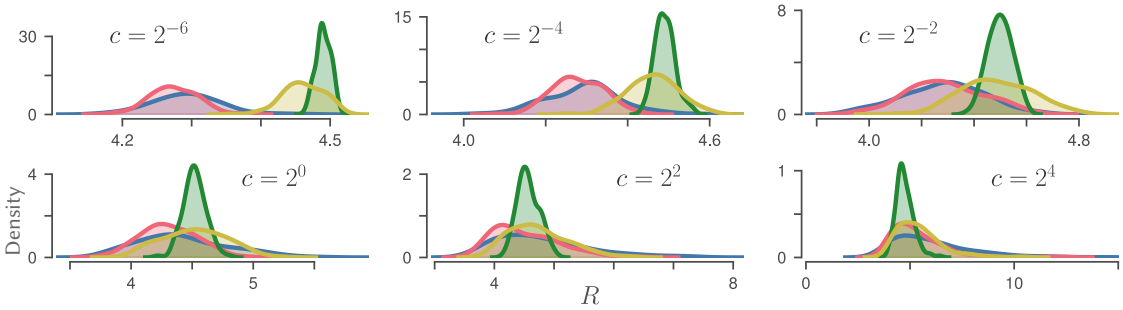


Figure 5.11.: R-distributions for the four candidate neuron models for different values of c.

different variations of data generated with equation (5.38): three levels of noise s (ranging from high, 2^{12}, to low 2^{20} [$\mathrm{m}^2 \cdot \mathrm{nm} \cdot \text{photons} \cdot \mathrm{sr} \cdot \mathrm{kW}^{-1}$]), three dataset sizes L (2^9, 2^{12}, and 2^{15}) and two wavelength ranges (20–1000 m and 15–30 m). Datasets with high noise and long wavelengths provide almost no discriminatory information: the two model predictions are almost the same, and any discrepancy between them is dwarfed by the amount of noise. At the other end, the low noise, short wavelength datasets are at the threshold between moderate and strong evidence in favour of $\mathcal{M}_\mathrm{P}$—in this regime the better model can already be identified by eye.

If Fig. 5.8 in the main text compares criteria on their terms, on a type of problem they were designed to tackle, Supplementary Table 5.3 explores what happens in a more challenging situation with misspecified models and an ambiguous selection problem. As with Fig. 5.8, all comparisons are done on the basis of a single test dataset, so replication uncertainty does not play a role.

In addition to the B^EMD and the risk (B^R), Supplementary Table 5.3 includes criteria based on the *likelihood ratio* (B^l), the *Bayes factor* (B^Bayes) [84, 134] and the *expected log predictive density* (elpd) (B^elpd) [56, 134]. We note that because $\mathcal{M}_\mathrm{P}$ and $\mathcal{M}_\mathrm{RJ}$ have the same number of parameters, the likelihood ratio is also equivalent to the *Akaike Information Criterion* (AIC), up to a factor of 2 [134].

To allow for comparability, for each criterion we report the log probability ratio; i.e. "model A is B^C_{AB} times more probable than model B." Conceptually, if $P(A)$ is the "probability of model A" and $P(B)$ the "probability of model B", then a criterion corresponds to

$$B^C_{AB} = \frac{P(A)}{P(B)} \quad \leftrightarrow \quad \log_{10} B^C_{AB} = \log_{10} P(A) - \log_{10} P(B)\,. \tag{5.86}$$

This is the typical form for Bayes' factors, and we use Jeffreys' scale [84, 191] to interpret values in Supplementary Table 5.3: values near 0, ±1 and ±2 respectively correspond to inconclusive, weak, and strong evidence.

Although only Bayes factors are typically reported in this form, most information criteria are defined in terms of log probabilities; the difference between the criteria of two models (the sign of which determines which model is selected) can be identified with the difference $\log_{10} P(A) - \log_{10} P(B)$ in equation (5.86).

The main exception is the EMD criterion, which is defined in terms of a probability $P(R_A < R_B)$ rather than separate probabilities for A and B. To express it as a probability ratio, we define the "underbar" quantity, obtained by applying a logit transformation to $B^\mathrm{EMD}_{AB;c}$:

$$\log_{10} \underline{B}^\mathrm{EMD}_{AB;c} \coloneqq \log_{10} B^\mathrm{EMD}_{AB;c} - \log_{10}\bigl(1 - B^\mathrm{EMD}_{AB;c}\bigr)\,. \tag{5.87}$$

Values of $\underline{B}^\mathrm{EMD}_{AB;c}$ are therefore not directly comparable with those of $B^\mathrm{EMD}_{AB;c}$ reported elsewhere in this paper. In Supplementary Table 5.3, the $\log_{10} \underline{B}^\mathrm{EMD}_{AB;c}$ ranges from $-\infty$ to ∞, with values $\pm\infty$ indicating that the two R-distributions have zero overlap.

As a comparison, we also define $\underline{B}^R$, to show what happens if we simply compare the true risk:

$$\log_{10} \underline{B}^R_{AB} \coloneqq (-R_A + R_B) \;/\; \log 10\,. \tag{5.88}$$

The division by $\log 10$ converts the basis of the logarithms from e to 10.

To interpret the results summarised in Supplementary Table 5.3, we highlight two types of undesirable behaviour:

Lack of saturation There should always come a point where enough data have been collected, and simply enlarging the dataset with samples from the same distribution does not provide more information. In the table therefore we want values to converge to a finite value as L increases.

Non-robustness The magnitude of a criterion should be an indicator of its robustness. If small changes in a dataset cause a criterion to flip from strongly negative to strongly positive, this suggests that its magnitude is not a good indication for the strength of the evidence.

If we are to interpret the values of criteria as relative probabilities, the global pattern we would expect to see in Supplementary Table 5.3 is that cells become progressively more blue as we go to the right (less noise, less model mismatch) and as we go down (more data, less ambiguous data).

Expressions used to compute selection criteria Conventions:

- $\mathcal{D}$: Training dataset used to fit the model.
- L: Number of data samples in $\mathcal{D}$.
- $(\lambda_i, \mathcal{B}_i)$: Sample in $\mathcal{D}$.
- $(\lambda'_j, \mathcal{B}'_j)$: Additional sample not in $\mathcal{D}$ (i.e. a test sample).
- $\hat{\sigma}, \hat{T}$: Estimates of σ and T obtained by maximizing the likelihood on $\mathcal{D}$.

EMD criterion

$$\log_{10} \underline{B}^{\mathrm{EMD}}_{AB;c} \coloneqq \log_{10} B^{\mathrm{EMD}}_{AB;c} - \log_{10}\bigl(1 - B^{\mathrm{EMD}}_{AB;c}\bigr)\,,$$

where $A = \mathcal{M}_{\mathrm{P}}$ and $B = \mathcal{M}_{\mathrm{RJ}}$ (repeated from equation (5.87)).

Loss function As we did for the neuron model, we define the loss of a point $(\lambda_i, \mathcal{B}_i)$ as the negative log likelihood given that data point. Since the candidate models assume Gaussian noise (equation (5.39)), this is simply

$$\begin{aligned} Q_a(\mathcal{B}_i \mid \lambda_i, \sigma, T) &= -\log p\Bigl(\mathcal{B} \;\Big|\; \mathcal{B}_a(\lambda; T), \sigma\Bigr) \\ &= \log\sqrt{2\pi}\sigma + \frac{(\mathcal{B} - \mathcal{B}_a(\lambda; T))^2}{2\sigma^2}\,. \end{aligned} \tag{5.89}$$

Here the subscript a is used to indicate whether we use predictions from the Rayleigh-Jeans (equation (5.36)) or Planck (equation (5.37)) model.

Log likelihood function Since we defined the loss as the negative log likelihood, one way to express the log likelihood for the whole dataset is simply as the negative total loss:

$$\ell_a(\sigma, T) \coloneqq \sum_{i=1}^{L} -Q_a(\mathcal{B}_i \mid \lambda_i, \sigma, T)\,. \tag{5.90}$$

Risk The risk is the expectation of the loss under the true model, so we have

$$R_a \coloneqq \Bigl\langle Q_a\Bigl(\mathcal{B}'_j \mid \lambda'_j, \hat{\sigma}_a, \hat{T}_a\Bigr)\Bigr\rangle_{\lambda'_j, \mathcal{B}'_j \sim \mathcal{M}_{\mathrm{true}}}\,, \tag{5.91}$$

$$\log_{10} \underline{B}^{R} \coloneqq \bigl(-R_{\mathrm{P}} + R_{\mathrm{RJ}}\bigr) \;/\; \log 10\,, \qquad \text{(repeated from (5.88))}$$

where $\lambda'_j, \mathcal{B}'_j \sim \mathcal{M}_{\mathrm{true}}$ indicates that λ'_j and $\mathcal{B}'_j$ are random variables with the same probability as the data.

Table 5.3.: Comparison of different model selection criteria for variations of the datasets shown in Figure 5.7. Criteria compare the Planck model ($\mathcal{M}_\mathrm{P}$) against the Rayleigh-Jeans model ($\mathcal{M}_\mathrm{RJ}$) and are evaluated for different dataset sizes (L), different levels of noise (s) and different wavelength windows (λ). To allow for comparisons, all criteria have been transformed into ratios of probabilities. **Values reported are the** $\log_{10}$ **of those ratios.** Positive (blue shaded) values indicate evidence in favour of the Planck model, while negative (red shaded) values indicate the converse. For example, a value of +1 is interpreted as $\mathcal{M}_\mathrm{P}$ being 10 times more likely than $\mathcal{M}_\mathrm{RJ}$, while a value of -2 would suggest that $\mathcal{M}_\mathrm{P}$ is 100 times *less* likely than $\mathcal{M}_\mathrm{RJ}$. Noise levels are reported in two ways: s is the actual value used to generate the data, while "rel. σ" reports the resulting standard deviation as a fraction of the maximum radiance within the data window. The 20–1000 µm window for λ stretches into the far infrared range, where the two models are nearly indistinguishable; hence higher positive values are expected for zero bias, low noise, and the 15–30 µm window. As in Figure 5.7, calculations were done for both positive and null bias conditions (resp. ($\mathcal{B}_0 > 0$ and $\mathcal{B}_0 = 0$); the former emulates a situation where neither model can fit the data perfectly. Expressions for all criteria are given in the supplementary text. For the $\underline{B}^\mathrm{EMD}_\mathrm{P,RJ}$ criteria, we used $c = 0.5$. Although this is the same value as we used for neuron model, it was determined through a separate calibration with different epistemic distributions.

			$\mathcal{B}_0 > 0$			$\mathcal{B}_0 = 0$		
		s	2^{12}	2^{16}	2^{20}	2^{12}	2^{16}	2^{20}
		rel. σ	35%	9%	2%	36%	9%	2%
Criterion	λ(m)	L						
$B^l_\mathrm{P,RJ}$	20–1000	512	−0.72	−0.74	5.64	−0.72	−0.82	4.62
		4096	0.64	−0.41	14.49	0.62	−1.05	7.94
		32768	−3.02	10.33	40.41	−3.23	8.11	−8.50
	15–30	512	0.06	−0.35	3.27	0.06	−0.39	2.61
		4096	0.52	3.86	52.20	0.49	3.53	47.12
		32768	3.04	23.02	389.62	2.87	20.33	348.85
$B^\mathrm{Bayes}_\mathrm{P,RJ;\pi}$	20–1000	512	0.02	−0.01	0.02	−0.06	0.04	-6×10^{-3}
		4096	0.06	−0.03	−0.02	0.04	−0.03	0.07
		32768	−0.05	−0.05	0.04	0.03	−0.03	−0.02
	15–30	512	-6×10^{-3}	−0.03	0.01	−0.02	-7×10^{-3}	−0.05
		4096	-4×10^{-3}	-5×10^{-3}	0.04	−0.04	−0.06	−0.01
		32768	0.02	0.06	−0.05	9×10^{-3}	0.05	−0.01
$B^\mathrm{elpd}_\mathrm{P,RJ;\pi}$	20–1000	512	-4×10^{-3}	3×10^{-5}	3×10^{-4}	2×10^{-4}	2×10^{-3}	5×10^{-4}
		4096	-9×10^{-4}	1×10^{-3}	-4×10^{-3}	-3×10^{-3}	-3×10^{-3}	7×10^{-4}
		32768	2×10^{-4}	2×10^{-3}	-2×10^{-3}	1×10^{-5}	-7×10^{-5}	-2×10^{-3}
	15–30	512	-4×10^{-3}	-2×10^{-3}	-2×10^{-3}	6×10^{-4}	2×10^{-3}	-4×10^{-4}
		4096	-3×10^{-3}	1×10^{-3}	6×10^{-4}	6×10^{-4}	-8×10^{-4}	-2×10^{-3}
		32768	6×10^{-4}	-2×10^{-4}	-1×10^{-3}	-2×10^{-3}	-2×10^{-3}	-1×10^{-3}
$\underline{B}^R_\mathrm{P,RJ}$	20–1000	512	2×10^{-5}	-5×10^{-4}	5×10^{-3}	2×10^{-5}	-7×10^{-4}	4×10^{-3}
		4096	2×10^{-5}	-5×10^{-4}	5×10^{-3}	2×10^{-5}	-7×10^{-4}	4×10^{-3}
		32768	2×10^{-5}	-5×10^{-4}	5×10^{-3}	2×10^{-5}	-7×10^{-4}	4×10^{-3}
	15–30	512	8×10^{-5}	6×10^{-4}	0.01	8×10^{-5}	6×10^{-4}	9×10^{-3}
		4096	8×10^{-5}	6×10^{-4}	0.01	8×10^{-5}	6×10^{-4}	9×10^{-3}
		32768	8×10^{-5}	6×10^{-4}	0.01	8×10^{-5}	6×10^{-4}	9×10^{-3}
$\underline{B}^\mathrm{EMD}_\mathrm{P,RJ}$	20–1000	512	−0.04	0.11	0.33	−0.07	0.12	0.41
		4096	−0.01	0.07	0.34	−0.01	0.07	0.39
		32768	-1×10^{-3}	−0.04	0.26	-1×10^{-2}	−0.05	0.29
	15–30	512	−0.01	0.31	3.08	0.05	0.34	3.50
		4096	−0.03	0.40	1.87	−0.02	0.39	1.80
		32768	−0.05	0.24	1.74	−0.04	0.25	1.66

Since in this case we know $\mathcal{M}_{\text{true}}$, the expectation can be computed exactly. We did this by generating a very large number of data samples ($L = 2^{12}$) and computing the empirical average of the loss:

$$R_a \approx -\frac{1}{L}\ell_a(\hat{\sigma}_a, \hat{T}_a)\,. \tag{5.92}$$

Bayesian prior The Bayesian calculations require a prior on the parameters σ and T. We used a simple prior that factorizes into two independent distributions:

$$\begin{aligned}\pi\Big(\log_2 \sigma\Big) &\sim \text{Unif}\Big(\Big[\log_2 2^9, \log_2 2^{14}\Big]\Big)\,,\\ \pi\Big(\log_2 T\Big) &\sim \text{Unif}\Big(\Big[\log_2 1000, \log_2 5000\Big]\Big)\,.\end{aligned} \tag{5.93}$$

Here T is expressed in Kelvin and σ has units $\text{kW}\cdot\text{m}^{-2}\cdot\text{nm}^{-1}\cdot\text{sr}^{-1}$. We chose log uniform distributions because these are more appropriate for parameters that are strictly positive and which can span multiple scales: the logarithmic scaling captures the fact that the difference between 1000 K and 1001 K is more significant than the difference between 5000 K and 5001 K. Likewise for differences in the parameter σ at opposite ends of its range.

Expected log pointwise posterior predictive density (elpd)

$$\text{elpd}_a \coloneqq \Big\langle \log_{10} p\big(\lambda'_j, \mathcal{B}'_j \mid a, \mathcal{D}\big)\Big\rangle_{\lambda'_j, \mathcal{B}'_j \sim \mathcal{M}_{\text{true}}} \tag{5.94}$$

$$\log_{10} B^{\text{elpd}} \coloneqq \text{elpd}_{\text{P}} - \text{elpd}_{\text{RL}}\,. \tag{5.95}$$

Note p in equation (5.94) is a posterior density, and therefore evaluating it involves integrating over the prior:

$$p\big(\lambda'_j, \mathcal{B}'_j \mid a, \mathcal{D}\big) = \iint d\sigma dT\, \pi_\sigma(\sigma)\pi_T(T)\; p\Big(\lambda'_j, \mathcal{B}'_j \mid a, \mathcal{D}, \sigma, T\Big)\,. \tag{5.96}$$

Relative likelihood and AIC The relative likelihood is given by

$$\log_{10} B^l \coloneqq \big(\ell_{\text{P}}(\hat{\sigma}_{\text{P}}, \hat{T}_{\text{P}}) - \ell_{\text{RL}}(\hat{\sigma}_{\text{P}}, \hat{T}_{\text{P}})\big) \;/\; \log 10\,, \tag{5.97}$$

while the difference between the AIC criteria of both models is (we use here the fact that both models have the same number of parameters)

$$\Delta AIC \coloneqq 2\ell_{\text{P}}(\hat{\sigma}_{\text{P}}, \hat{T}_{\text{P}}) - 2\ell_{\text{RL}}(\hat{\sigma}_{\text{P}}, \hat{T}_{\text{P}})\,. \tag{5.98}$$

Since the two are equivalent up to a factor $2\log 10$, the trends we see with the likelihood ratio therefore also occur with the AIC.

(The factor 2 in the AIC is meant to allow it to be interpreted—under certain assumptions—as a draw from a χ^2 distribution. This correspondence however is not required when interpreting Supplementary Table 5.3.)

Model evidence The model evidence is used to compute the Bayes factors. It is the expectation of the likelihood of the data—$\mathcal{D} = \{\lambda_i, \mathcal{B}_i\}_{i=1}^L$—under the prior for T and σ:

$$\begin{aligned}\mathcal{E}_a &\coloneqq \iint d\sigma dT\, \pi_\sigma(\sigma)\pi_T(T)\; p\Big(\{\lambda_i, \mathcal{B}_i\}_{i=1}^L \mid a, \sigma, T\Big)\\ &= \iint d\sigma dT\, \pi_\sigma(\sigma)\pi_T(T)\ell_a(\sigma, T)\;\;.\end{aligned} \tag{5.99}$$

The likelihood $p\Big(\{\lambda_i, \mathcal{B}_i\}_{i=1}^L \mid a, \sigma, T\Big)$ is given by the Gaussian observation model.

Bayes factor

$$\log_{10} B^B \coloneqq \log_{10} \mathcal{E}_{\text{P}} - \log_{10} \mathcal{E}_{\text{RL}}\,. \tag{5.100}$$

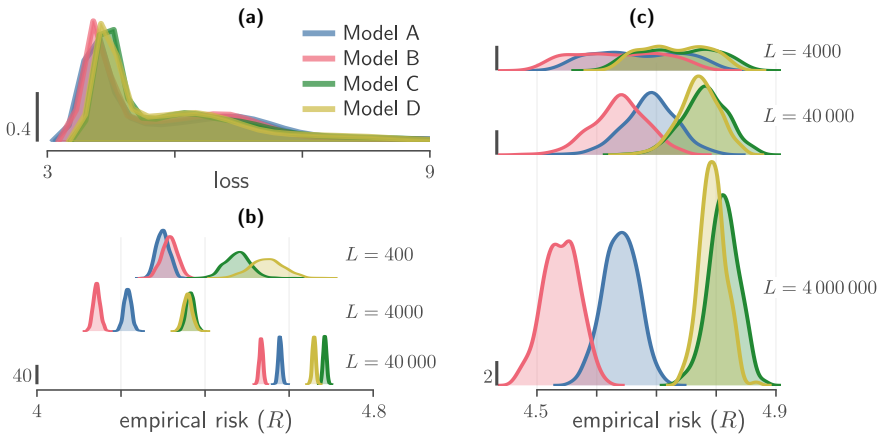


Figure 5.12.: Aleatoric and finite-size uncertainty. a) Loss distribution of individual data points—$\{Q_a(t_k, V^{\mathrm{LR}}(t_k; \mathcal{M}_{\mathrm{true}}))\}$—for the dataset and models shown in Fig. 5.2 and $a \in \{A, B, C, D\}$. For a model that predicts the data well, this is mostly determined by the aleatoric uncertainty. **b) Bootstrap estimate of finite-size uncertainty** on the risk, obtained using case resampling [192]: for each model, the set of losses was resampled 1200 times with replacement. Dataset and colours are the same as in (a); the same L data points are used for all models. **c) Synthetic estimate of finite-size uncertainty** on the risk, obtained by evaluating equation (5.10) on 400 different simulations of the candidate model (differing by the random seed). Here the same model is used for simulation and loss evaluation. Dataset sizes L determine the integration time, adjusted so all datasets contain the same number of spikes. All subpanels use the same vertical scale. **b–c)** The variance of the R-distributions, i.e. the uncertainty on R, goes to zero as L is increased. **a–c)** Colours indicate the model used for the loss. Probability densities were obtained by a kernel density estimate (KDE).

5.C. Supplementary Discussion

5.C.1. Other forms of uncertainty

As we say in the main text, there are two main sources of epistemic uncertainty on an estimate of the risk: limited number of samples and variability in the replication process. This work focusses on the estimating replication uncertainty, and numerically studying its effect as a function of sample size. We have eschewed a formal treatment of sample size effects, since this is a well-studied problem and good estimation procedures already exist.

For example, a bootstrap procedure can be used to estimate the uncertainty on a statistic (here the risk R) from a single dataset $\mathcal{D}$, by recomputing the statistic on multiple surrogate datasets obtained by resampling $\mathcal{D}$ with replacement (Supplementary Fig. 5.12**b**). Alternatively, if we have access to good candidate models, we can use those models as simulators to generate multiple synthetic datasets (Supplementary Fig. 5.12**c**). The distribution of risks over those datasets is then a direct estimate of its uncertainty due to finite samples.

In the limit of infinite data, both of these methods produce a risk "distribution" that collapses onto a precise value, independent of any discrepancy between model and true data-generating process. In contrast, that discrepancy defines the spread of risk distributions in Fig. 5.3, which do not collapse when $L \to \infty$.

Aleatoric uncertainty also does not vanish in the large L limit, but manifests differently. It sets a lower bound on the spread of pointwise losses a model can achieve (Supplementary Fig. 5.12**a**). In terms of our formalism therefore:

- **aleatoric uncertainty** determines the shape of the PPFs q^* and $\tilde{q}$,
- **epistemic uncertainty (due to finite samples)** is the statistical uncertainty on those shapes, and
- **epistemic uncertainty (across replicates)** is the propensity of the PPF to change when the experiment is replicated.

Both forms of epistemic uncertainty can contribute uncertainty on the risk, i.e. increase the spread of the R-distribution, but in the $L \to \infty$ limit only the effect of replications remains. Changes to aleatoric uncertainty will shift R-distributions along the R axis, but do not directly contribute epistemic uncertainty.

5.C.2. Flexibility in selecting c

An important property of our approach is that sensitivity parameter c does not need to be tuned to a precise value, but can lie within a range; either $c \in [2^{-4}, 2^{0}]$ for the neuron models of Fig. 5.6, or $c \in [2^{-6}, 2^{3}]$ for the Planck and Rayleigh-Jeans models of Fig. 5.9. This does not mean that the value of $B^{\mathrm{EMD}}_{ab;c}$ itself is insensitive to c: for fixed data, a larger c will generally bring $B^{\mathrm{EMD}}_{ab;c}$ closer to 50%. But the probability bound given by $B^{\mathrm{EMD}}_{ab;c}$ remains correct for all c within that range.

For example, we compute $B^{\mathrm{EMD}}_{\mathrm{P,RJ};c}$ to be 80% when $c = 2^{-3}$, versus 60% when $c = 2^{0}$. This means that if we fix $c = 2^{-3}$ (resp. $c = 2^{0}$), among all experiments that yield $B^{\mathrm{EMD}}_{\mathrm{P,RJ};c} = 80\%$ (resp. $B^{\mathrm{EMD}}_{\mathrm{P,RJ};c} = 60\%$), in at least 80% (resp. 60%) of them model $\mathcal{M}_{\mathrm{P}}$ will have lower true risk than $\mathcal{M}_{\mathrm{RJ}}$.

More generally, a $B^{\mathrm{EMD}}_{ab;c} = B$ (with $0.5 < B \leq 1$) states that, of all the experiments where the calculation of $B^{\mathrm{EMD}}_{ab;c}$ is equal to B, at least a fraction B of those will have $R_a < R_b$. In this respect the B^{EMD} exhibits some similarities with a confidence interval, in that it is interpreted in terms of *replications under a fixed computational procedure*: in the former case we have a fixed procedure for calculating $B^{\mathrm{EMD}}_{ab;c}$ given c, while in the latter case we have a fixed procedure for calculating the confidence interval given a confidence level. A key difference however is that with the B^{EMD}, the interpretation requires conditioning on the outcome of the calculation.

Of course the range of valid c values will depend on the variety of epistemic distributions, and cannot be guaranteed. In general, the larger the variety, the more difficult one can expect it to be to find a c that is valid in all conditions. We can however anticipate a few strategies that might increase the range of valid c values and otherwise improve the ability of the B^{EMD} criterion to discriminate between models:

Increased rejection threshold Increasing the rejection threshold ϵ in equation (5.15) can be a simple way to add a safety margin to the B^{EMD} criterion, at the cost of some statistical power, to account for small violations of equation (5.35).

Multi-step comparisons Initially, the large number of candidate models may force the selection of a larger (i.e. more conservative) c. After using this c to reject some of the candidates, it may be possible to reduce the value of c, thus increasing the discriminatory power and possibly further reducing the remaining pool of candidates.

Improved experimental control If one can improve the reproducibility across experiments, the epistemic distributions used for calibration can correspondingly be made tighter. This makes it easier to find a c that works in all experimental conditions, since there are overall fewer conditions to satisfy.

Domain-informed loss function A loss function designed with knowledge of the domain or target application can ignore irrelevant differences between models. In addition to improving the relevance of comparisons, this tends to make the risk a smoother function of experimental parameters, which should make the B^{EMD} easier to calibrate.

Post hoc correction of B^{EMD} values As long as we select a c for which the $B^{\mathrm{epis}}_{AB;\Omega}\left(B^{\mathrm{EMD}}_{AB;c}\right)$ function of equation (5.54) is monotone, we can use the calibration curves themselves to interpret values of $B^{\mathrm{EMD}}_{AB;c}$ by looking to the $B^{\mathrm{epis}}_{AB;\Omega}$ to which they map. This approach could be used to improve discriminatory power of a conservative B^{EMD}, but also to correct it in regions where it is overconfident—assuming one has sufficient trust that the calibration curves are truly representative of experimental variations.

5.C.3. Comparing models directly with the loss distribution

To further illustrate the previous point, we consider an alternative comparison criterion that directly uses the distribution of losses Q (i.e. the distribution in Supplementary Fig. 5.12**a**) instead of the more complicated process $\mathfrak{Q}$ over PPFs with which we defined the B^{EMD}. To this end, let us define a B^{Q} criterion in analogy with equation (5.16):

$$\begin{aligned} B^{Q}_{ab;c_Q} &:= P(Q_a < Q_b + \eta)\,, \\ \eta &\sim \mathcal{N}(0, c_Q^2)\,. \end{aligned} \tag{5.101}$$

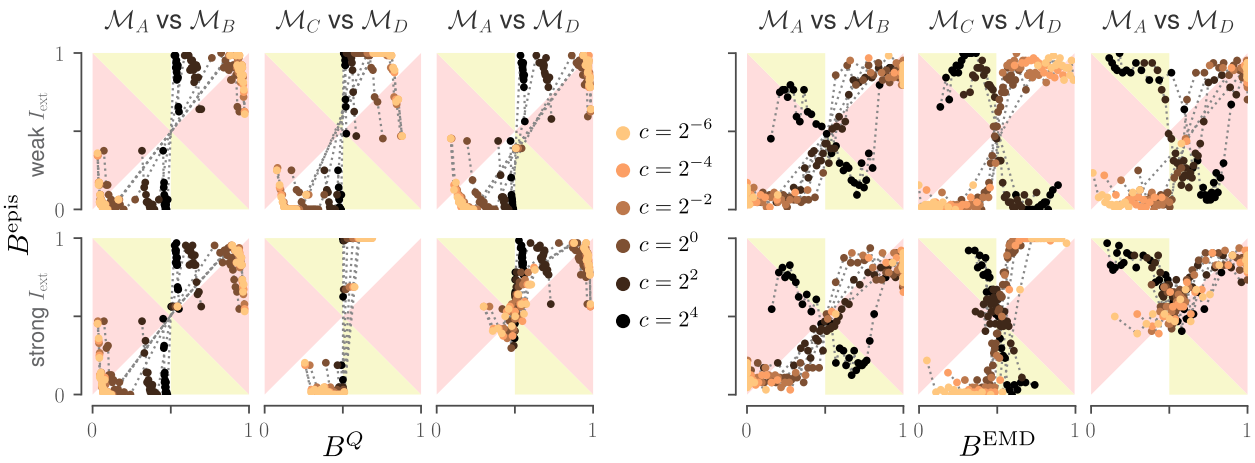


Figure 5.13.: (a) Calibration experiments for the neuron model, using B^{Q} (equation (5.101)) instead of B^{EMD} (equation (5.16)) as a comparison criterion. To better show the distribution of experiments, each histogram bin (see Calibration experiments in the Methods) is represented as a point. **(b)** Same curves as in Fig. 5.6, this time with bins presented as points to ease comparison with (a). Compared to (a), points are more uniformly distributed along both the horizontal and vertical axes. All panels use the same set of sensitivity values (either c_Q or c), with colours as indicated in the central legend.

The Gaussian noise η is added to allow us to adjust the sensitivity of the criterion, analogously to how c adjusts the sensitivity of the $B^{\text{EMD}}_{ab;c}$ criterion. Both criteria thus have one free parameter, making the comparison relatively fair.

Note that in contrast to the B^{EMD}, the c_Q values in the B^{Q} are not dimensionless. Consequently, they would likely be less transferable between experimental contexts.

A possible argument in favour of using B^{Q} as a criterion is that increasing the amount of misspecification—for example by increasing the unmodelled bias $\mathcal{B}_0$ in equation (5.38)—will affect the distribution of losses in some way. So although the shape of the PPFs is mostly determined by aleatoric uncertainty (as evidenced by the similarity of the distributions in Supplementary Fig. 5.12**a**), one can expect them to also contain some information about the amount of misspecification—and thereby also the amount of epistemic uncertainty.

Nevertheless, as we see in 5.13 , the B^{Q} criterion is less effective for comparing models than the B^{EMD} in at least three ways (recall that the goal is not to find a criterion that accurately predicts the model with lowest risk, but one that accurately predicts the *uncertainty* on that risk, i.e. B^{epis}):

Reduced signal Only very few experiments on the main diagonal have B^{epis} values different from 0 or 1. The B^{Q} values therefore don't really inform about the epistemic uncertainty. Learning a monotone transformation from B^{EMD} or B^{Q} to B^{epis} is only possible if the curve is strictly monotone.

Increased anti-correlation between B^{Q} and B^{epis}: in five out of six comparisons between neuron models, we see strong anticorrelated tails at both ends of the curves, which dip all the way back to $B^{\text{epis}} \approx 0.5$. This makes it harder to find a c_Q for which the the B^{Q} is actually informative, since the curve is not even injective. While there is some anticorelation also with the B^{EMD} (5.13**b**), it is much less pronounced and limited to comparisons with the worse models ($\mathcal{M}_C$ and $\mathcal{M}_D$).

The sensitivity parameter does not really help Increasing c_Q squeezes curves horizontally, bringing all B^{Q} values closer to 0.5, but does not change their shape nor reduce the length of anticorrelated tails at both ends. All curves therefore effectively provide the same information. In contrast, the c parameter has a much stronger effect on the shape of R-distributions (c.f. Supplementary Fig. 5.11).

These results suggest that an approach based only on distributions of the loss Q is likely to always struggle with disentangling epistemic from aleatoric uncertainty. It also has less information to work with: whereas B^{EMD} is parameterised by two functions, q^* and δ^{EMD}, B^{Q} is parameterised by only one, the density $p(Q)$. Moreover, $p(Q)$ and q^* contain the same information (they are the PDF and PPF of the same random variable), so the discrepancy

function δ^{EMD} provides B^{EMD} with strictly more information—information which is also specifically designed to disentangle the effects of misspecification from aleatoric uncertainty.

Conclusion

Tools of the future

One often underappreciated aspect of the kind of work presented in the previous chapters is the degree to which it depends on good computational tools: As models become more complex, these tools need to improve alongside, to allow us to define models at a increasingly high-levels. Otherwise our focus is increasingly spent on just getting models to run, and we run out of bandwidth to actually make the science progress.

> Better tools free your mind. —David Duvenaud [193]

The `sinn` library [194] built for inferring the mesoGIF model (Chapter 4) was a step in this direction, but remained too experimental to gain wider traction. Within that project however, it undoubtedly demonstrated its value, and for that reason I remain convinced that someday we will see published a library specifically for learning interpretable models.

Another surprising challenge as models and analysis pipelines become more complex is simply keeping track of experiments: with dozens of possible modelling decisions—and the computational ability to actually execute a substantial fraction of them—this is far from from a trivial problem. Not only is effort spent logging experiments effort not spent on the science, but with increased logging overhead also comes increased probability of logging errors that can contaminate the science.

My `SumatraTask` [195] software addresses this by extending an existing provenance tracking project (`Sumatra` [196]) with a more flexible interface. I continue to find it valuable for managing my computational experiments, by automating the task of tracking and retrieving collections of experiments. For example, it considerably simplified the task of managing the thousands of calibration experiments of Section 5.2.7.

These kinds of tools will become essential if we are to translate the methods developed in this thesis, which were all at the demonstration level, into methods used in practice: Widespread adoption requires that an analysis pipeline like that illustrated in Figure 5.1 essentially run out of the box. Although a general purpose library for EMD is available [158], and we did test the method to the best of our abilities, true validation of the method will require other research groups to integrate it into their methodologies, and find that it works for them.

We should look forward to a time when comparing fitted models based on their replicability becomes commonplace, as this would bring modelling research more in line with the principles of scientific induction. Principles that allow us not only to ask new questions, but also to provide more definitive answers. In our case this would mean more confidently pruning a pool of candidate models fitted with backpropagation through time, as in Chapter 4. We imagine cases where candidate pools are pruned to the point that all retained models implement the same mechanism—for example, they might share a bifurcation structure.

This I hope is the future for this line of work: to enable the data-driven discovery of new mechanism in complex dynamical systems.

Beyond neuroscience

The work in this thesis has rested on three methodological pillars: interpretable neuroscience models, machine learning, and model selection. Each of these has its own scientific tradition, built on decades of research and uncountable trials and errors. Those traditions can sometimes seem outright incompatible: bringing them together involves substantial efforts of translation and often some adaptation.

This is not to say that it cannot be done. Indeed, as the previous chapters have shown, there is the potential for real scientific breakthroughs when we bring to bear our day's large and accessible computing power on hard modelling problems. However we must stay vigilant, resist the swan song of anecdotal confirmation, and keep our tradition of always questioning and testing our theories—lest they become built as houses of cards.

Of course there are other pillars in science, and thus many more opportunities for fruitful synergies. And especially with the central role that machine learning continues to play in all spheres of research, we can expect the techniques developed here to appear and grow beyond our own garden of neuroscience.

Software

Pyloric network simulator

A.

This simulator is used for most of the experiments of Chapter 5.
The implementation is published as [197] and is available on the Python Package Index at https://pypi.org/project/pyloric-network-simulator/

The pyloric circuit model described by Prinz at al.[15, 21] continues to be studied today, both as an example conductance model of an interesting biological process and as a case study of functional diversity in computational models [24]. It is used as a case study for both the *Brian 2* [198] and the `xolotl` [199] simulators; the original C++ implementation can be found on *ModelDB* [200] and has been ported to the *OpenSourceBrain* [201]

This chapter describes a reimplementation of the model in pure Python, using JAX [202] to achieve comparable execution speeds as C/C++. This makes it especially suitable for users who want to modify the code themselves, or integrate as part of a Python stack.

The main distinguishing features of this feature are:

JAX-accelerated Python implementation Combines the flexibility of Python with the speed of compiled JAX code.

The specification of update equations for ion channels was carefully designed to maximize vectorization.

Choice of ODE solver The official C++ implementation [200] uses hard-coded Euler integration, which can make it vulnerable to numerical instabilities. (Spiking neuron conductance-based models, after all, are stiff by design.)

In contrast, this implementation is designed to be used with Python's ODE solvers, making it easy to use a more appropriate solver. In particular, the default is to use an adaptive step solver, which can dramatically reduce simulation time while still keeping enough time precision to resolve spikes.

Modularity New cell types are defined simply by extending Python arrays for the ion channel parameters. If needed, an arbitrary function can also be given for specific ion channels.

Flexibility Use is not limited to the published three-cell network: studying a smaller two-cell network, or a larger one with different LP cells, is a simple matter of changing three arrays.

All-in-one documentation The original specification of the pyloric circuit model is spread across at least three resources [12, 15, 21].

Here all definitions are included in the inlined documentation, are fully referenced and use consistent notation.

Single source of truth Both the documentation and the code use the same source for parameter values, which ensures that the documented values are actually those used in the code.

On-demand thermalization (see Section A.4.1 (p. 162)) The original implementation performs an initial sweep over 20 million parameter sets for individual neurons, integrating each for a long period (~ 5 min simulated time)

in order to build a database of realistic states. This database was quite large (a few GB), and was therefore distributed separately from the code. It also meant that subsequent simulations are limited to one of the pre-computed states.

To the best of our understanding, the other published implementations provide the equations and parameters, but not the thermalized states.

Here models are thermalized automatically the first time a new parameter combination is used and the result saved to an on-disk cache. This makes thermalization completely transparent to the user (modulo a 5–10 minute wait time), ensures that it is performed only on those models that we actually need, and thus allows more fine-grained parameter exploration than what is possible via grid search.

A.1. Usage example

```
from pyloric_network_simulator.prinz2004 import \
    Prinz2004, neuron_models, dims
```

Instantiate a new model by specifying:

- The number of neurons of each type (PD, AB, LP and PY).
- The connectivity `gs` between populations.
  A value of 0 indicates no connection between the corresponding types.
- The set of membrane conductances for each type `g_cond`.
  The 16 sets that define the neuron models used in Prinz, Bucher, and Marder [21] are provided as a Pandas DataFrame in the variable `neuron_models`. Their values are listed in Table A.4 (p. 157) below.

```
model = Prinz2004(
    pop_sizes = {"PD": 2, "AB": 1, "LP": 1, "PY": 5},
    gs = [ [    0  ,     0  ,     3  ,     0 ],
           [    0  ,     0  ,     3  ,     3 ],
           [    3  ,     3  ,     0  ,     3 ],
           [    3  ,     3  ,     3  ,     0 ] ],
    g_ion = neuron_models.loc[["AB/PD 1", "AB/PD 1",
                               "LP 1", "PY 1"]]
)
```

Listing A.1: Note: The two subarrays of cholinergic (PD) and glutamatergic (AB, LP, PY) neurons are contiguous, to allow more efficient indexing based on slices. **Hint:** `pop_size` keys should match one of the four type labels (AB, PD, LP, PY) or the combined label "AB/PD". If there are multiple neurons of one type, they can be followed by a number: "AB 1", "AB 2", ... `g_ion` may be specified as Pandas DataFrame or NumPy Array.

Define a set of time points and evaluate the model at those points. A model simulation is always initialized with the result of a cached warm-up simulation (see Initialization). If no such simulation matching the model parameters is found in the cache, it is performed first and cached for future runs.

The example below reproduces the procedure of Prinz, Bucher, and Marder [21]: After neurons are connected, an initial simulation of 3s is thrown away to let transients decay. Then 1s is simulated at a resolution of 1ms per bin to generate the data for that network model.

Listing A.2: Evaluating the model at 1001 sequential time points, from 3000–4000. Values are in milliseconds. **Note:** The time points we provide to `model` are only those that are recorded; they have no bearing on the integration time step.[a] To resolve spikes, a time resolution of 1 ms or less is needed.

```
res = model(np.linspace(3000, 4000, 1001))
```

a: Integration is performed using a Runge-Kutta 4(5) scheme with adaptive time step, which can number in the 100's or more time steps per millisecond. If we recorded all of these we would quickly exceed memory limits.

Results are returned as a `SimResult` (p. 168) object, which has attributes to retrieve the different traces: membrane potential `V`, calcium concentration `Ca`, synaptic activation `s`, membrane activation `m` and membrane inactivation `h`. These are returned as Pandas DataFrames.

```
Vtraces = res.V.loc[:,[("AB", 1), ("LP", 1), ("PY", 1)]] \
              .droplevel("index", axis="columns")
```

Listing A.3: Retrieving trace for one neuron per population.

Symbol	Meaning	Dimensions
`res.V`	Membrane potential	time bins × neurons
`res.Ca`	Intracellular calcium concentration	time bins × neurons
`res.s`	Synapse activation	time bins × neurons
`res.m`	Channel activation	time bins × channels × neurons
`res.h`	Channel inactivation	time bins × channels × neurons
`res.t`	Time bin array	time bins

Table A.1.: Time-dependent variables retrievable from `SimResult`

To inspect the initialization curves, we use the private method `thermalize()` (p. 169); this is used internally to generate the thermalized state. It returns a `SimResult` (p. 168) object.

```
res = model.thermalize()
Vtraces = res.V.droplevel("index", axis="columns")
```

Listing A.4: Retrieving traces from the thermalization simulation.

Thermalization takes an unrealistic state of channel activations as initial condition and lets it relax to a more biophysically plausible state. This used to determine a sensible initial condition for simulations on the circuit. Populations are disconnected during thermalization, so only one neuron per population needs to be simulated; we can therefore drop the `"index"` dimension since it becomes redundant.

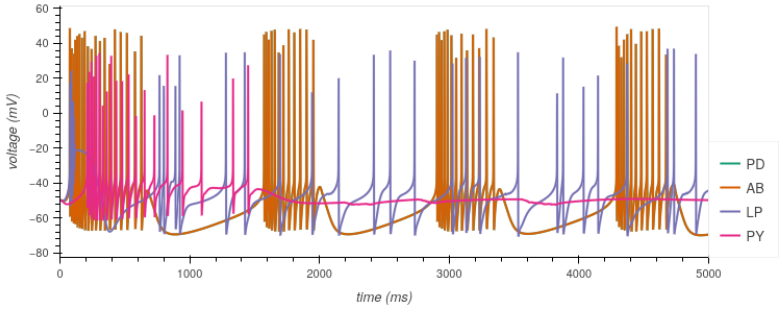


Figure A.1.: Thermalization of pyloric neurons models Output of Listing A.4, showing the thermalization of models `AB 1`, `PD 1`, `LP 1` and `PY 1`. (Curves `AB` and `PD` are overlayed since the models are identical.) In this case, initial transients decay in about 2 s.

A.2. Single compartment conductance model

As described in Section 1.2 (p. 15), each cell type is modelled with six ion-specific channels: one sodium channel (I_{Na}), two calcium channels (I_{CaT}, I_{CaS}), three potassium channels (I_{A}, $I_{\mathrm{K(Ca)}}$ and I_{Kd}). Two additional currents are also modelled: a "hyperpolarization-activated" channel is permeable to both Na^+ and K^+ and produces the current I_H, and a catch-all leak current I_{leak} models any remaining currents.

For ease of reference, we reproduce the key dynamical equations; for definitions, see Equations (1.17) and (1.21) (p. 12) and (p. 17) in Section 1.2 (p. 15).

$$\begin{aligned} \frac{C}{A}\frac{dV}{dt} &= -\sum_i I_i - I_{\text{input}} \\ I_i &= g_i m_i^p h_i (V - E_i) \\ I_{\text{input}} &= I_e + I_s + I_{\text{ext}}\,, \quad \tau_m \frac{dm}{dt} = m_\infty - m \\ \tau_h \frac{dh}{dt} &= h_\infty - h\,. \end{aligned} \tag{A.1}$$

Parameter values are given in are given in Tables A.2 to A.4 (pp. 156–157).

Recall that in this model the Ca^{2+} channels are concentration-dependent, so that we also need to keep track of the intracellular calcium concentration to compute the reversal potential E_{Ca} (reproduced from Equations (1.22) and (1.23) (p. 17) and (p. 18)):

$$\tau_{Ca}\frac{d[Ca^{2+}]_{in}}{dt} = -f(I_{CaT} + I_{CaS}) - [Ca^{2+}]_{in} + [Ca^{2+}]_0\,,$$

$$E_{Ca} = \underbrace{\frac{kT}{ze}}_{\gamma} \ln \frac{[Ca^{2+}]_{out}}{[Ca^{2+}]_{in}} = \gamma \ln \frac{[Ca^{2+}]_{out}}{[Ca^{2+}]_{in}} \quad [\text{mV}]$$

Ensuring positive concentrations

Although a concentration must always be positive, the differential equation for $[Ca^{2+}]$ reproduced here from Prinz, Bucher, and Marder [21] does not *per se* prevent the occurrence of a negative concentration. (Sustained CaT and CaS currents could drive $[Ca^{2+}]$ below zero.) In practice the rest of the ODE dynamics seem to prevent this, but nevertheless to ensure $[Ca^{2+}]$ is always positive and improve numerical stability, in our implementation we track $\log[Ca^{2+}]$ instead. Since concentrations can span multiple orders of magnitude, considering them in log space is in fact rather natural. A similar thing can be said of m and h, which must be bounded within $[0, 1]$; in this case we use a logit transformation to ensure the variables never exceed their bounds. We do the same (p. 160) for the synapse activations.

A.2.1. Constant low-dimensional parameters

Since lobsters are ectotherms (cold-blooded) and live in cold shallow Atlantic waters, a reasonable temperature range might be 1–10°C, which corresponds to the range $\gamma \in (11.8, 12.2)$ mV for ions with $z = 2$. In this implementation we fix γ to 12.2 mV,[1] and following [15], further set $[Ca^{2+}]_{out}$ to 3 mM.

1: This matches the value for `Nernstfactor` is the published source code. https://biology.emory.edu/research/Prinz/database-sensors/#download

If we multiply all the units in Equation (A.1) together, using 10^{-3} cm^2 for the unit of A, we find that they simplify to 1 mV/ms – the desired units for dV/dt. Therefore we can write the implementation using only the magnitudes in the *values* column of Table A.3 and ignore the units. We also omit C and A since their magnitudes cancel.

	I_{Na}	I_{CaT}	I_{CaS}	I_A	$I_{K(Ca)}$	I_{Kd}	I_H	I_{leak}	unit
E_i	50	E_{Ca}	E_{Ca}	-80	-80	-80	-20	-50	mV
p	3	3	3	3	4	4	1		

Table A.2.: Ion channel parameters for the dynamics defined in Equation (A.1). Values are taken from Prinz, Billimoria, and Marder [15].

Constant	Value	Unit
A (membrane area)	0.628	10^{-3} cm^2
C	0.628	nF
τ_{Ca}	200	ms
f	14.96	µM/nA
$[Ca^{2+}]_0$	0.05	µM
$[Ca^{2+}]_{out}$	3000	µM
γ	12.2	mV
approximate numerical time step	0.025	ms

Table A.3.: Constants used in Equation (A.1)

A.2.2. Maximum channel conductances

Prinz, Billimoria, and Marder [15] performed a sweep of possible values for the maximal conductance, identifying different sets of conductance values g_i for which Equation (A.1) qualitatively matches experiments. They reduced the values of experimentally-compatible conductances to 5–6 qualitatively different sets per neuron type, which are listed in Table A.4. Crucially, values must be taken as fixed sets; for example, interpolating between two sets for LP neurons will *not* yield a neuron model that is compatible with the observed activity of an LP neuron.

Table A.4.: Maximal conductance densities of model neurons, reproduced from Table 2 in Prinz, Bucher, and Marder [21]. Differences in these values are what differentiates neuron models.

	Maximal membrane conductance (g) in mS/cm^2							
Model neuron	$g(I_{Na})$	$g(I_{CaT})$	$g(I_{CaS})$	$g(I_A)$	$g(I_{K(Ca)})$	$g(I_{Kd})$	$g(I_H)$	$g(I_{leak})$
AB/PD 1	400	2.5	6	50	10	100	0.01	0.00
AB/PD 2	100	2.5	6	50	5	100	0.01	0.00
AB/PD 3	200	2.5	4	50	5	50	0.01	0.00
AB/PD 4	200	5.0	4	40	5	125	0.01	0.00
AB/PD 5	300	5.0	2	10	5	125	0.01	0.00
LP 1	100	0.0	8	40	5	75	0.05	0.02
LP 2	100	0.0	6	30	5	50	0.05	0.02
LP 3	100	0.0	10	50	5	100	0.00	0.03
LP 4	100	0.0	4	20	0	25	0.05	0.03
LP 5	100	0.0	6	30	0	50	0.03	0.02
PY 1	100	2.5	2	50	0	125	0.05	0.01
PY 2	200	7.5	0	50	0	75	0.05	0.00
PY 3	200	10	0	50	0	100	0.03	0.00
PY 4	400	2.5	2	50	0	75	0.05	0.00
PY 5	500	2.5	2	40	0	125	0.01	0.03
PY 6	500	2.5	2	40	0	125	0.00	0.02

A.2.3. Voltage-dependent activation variables

To implement the dynamics of the activation variables in code, we write them as

$$x(V, [Ca^{2+}]; y, a, b, c, C) = y(V, [Ca^{2+}])(\sigma(V; a, b, c) + C)$$
$$\sigma(V; a, b, c) = \frac{a}{1 + \exp\left(\frac{V+b}{c}\right)}, \tag{A.2}$$

where x is one of m_∞, h_∞, τ_m or τ_h. This form allows us to implement them as almost entirely vectorized operations, which calculate all activation variables simultaneously. Only the $y(V, [Ca^{2+}])$ function requires a loop over the five channel types for which it is non-zero.

The resulting curves are shown in Figure A.2.

Note

- When $a = 0$, the values of b and c are irrelevant, but we should not use $c = 0$ to avoid dividing by zero.
- The inactivating variable h is not used for $I_{K(Ca)}$, I_{Kd} and I_H. Because we use vectorized operations, the computations are still performed, but the result is afterwards discarded.
- The variable $\tau_m(I_H)$ is defined purely in terms of y. We do this by setting $\sigma = 0$ and $C = 1$.

Table A.5.: Voltage dependence – parameters a, b, c and C in Equation (A.2)

	m_∞				h_∞				τ_m				τ_h			
	a	b	c	C	a	b	c	C	a	b	c	C	a	b	c	C
I_{Na}	1	25.5	−5.29	0	1	48.9	5.18	0	−2.52	120	−25	2.64	1	34.9	3.6	1.5
I_{CaT}	1	27.1	−7.2	0	1	32.1	5.5	0	−42.6	68.1	−20.5	43.4	−179.6	55	−16.9	210
I_{CaS}	1	33	−8.1	0	1	60	6.2	0	0	0	1	1	0	0	1	1
I_A	1	27.2	−8.7	0	1	56.9	4.9	0	−20.8	32.9	−15.2	23.2	−58.4	38.9	−26.5	77.2
$I_{K(Ca)}$	1	28.3	−12.6	0	0	0	1	1	−150.2	46	−22.7	180.6	0	0	1	1
I_{Kd}	1	12.3	−11.8	0	0	0	1	1	−12.8	28.3	−19.2	14.4	0	0	1	1
I_H	1	75	5.5	0	0	0	1	1	0	0	1	1	0	0	1	1

The units for the denominator of $m_\infty[I_{K(Ca)}]$ were not reported in the original publications. However other equations for $[Ca^{2+}]$ are given in µM, and if we check the original source code [200] we can see that these are the correct units here as well.

Table A.6.: Voltage dependence - dynamic variable $y(V, [Ca^{2+}])$ in Equation (A.2)

	m_∞	h_∞	τ_m	τ_h
I_{Na}				$\frac{1.34}{1+\exp\left(\frac{V+62.9}{-10}\right)}$
I_{CaT}			$2.8+\frac{14}{\exp\left(\frac{V+27}{10}\right)+\exp\left(\frac{V+70}{-13}\right)}$	$120+\frac{300}{\exp\left(\frac{V+55}{9}\right)+\exp\left(\frac{V+65}{-16}\right)}$
I_{CaS}				
I_A				
$I_{K(Ca)}$	$\frac{[Ca^{2+}]}{[Ca^{2+}]+3\,\mu M}$			
I_{Kd}				
I_H			$\frac{2}{\exp\left(\frac{V+169.7}{-11.6}\right)+\exp\left(\frac{V-26.7}{14.3}\right)}$	

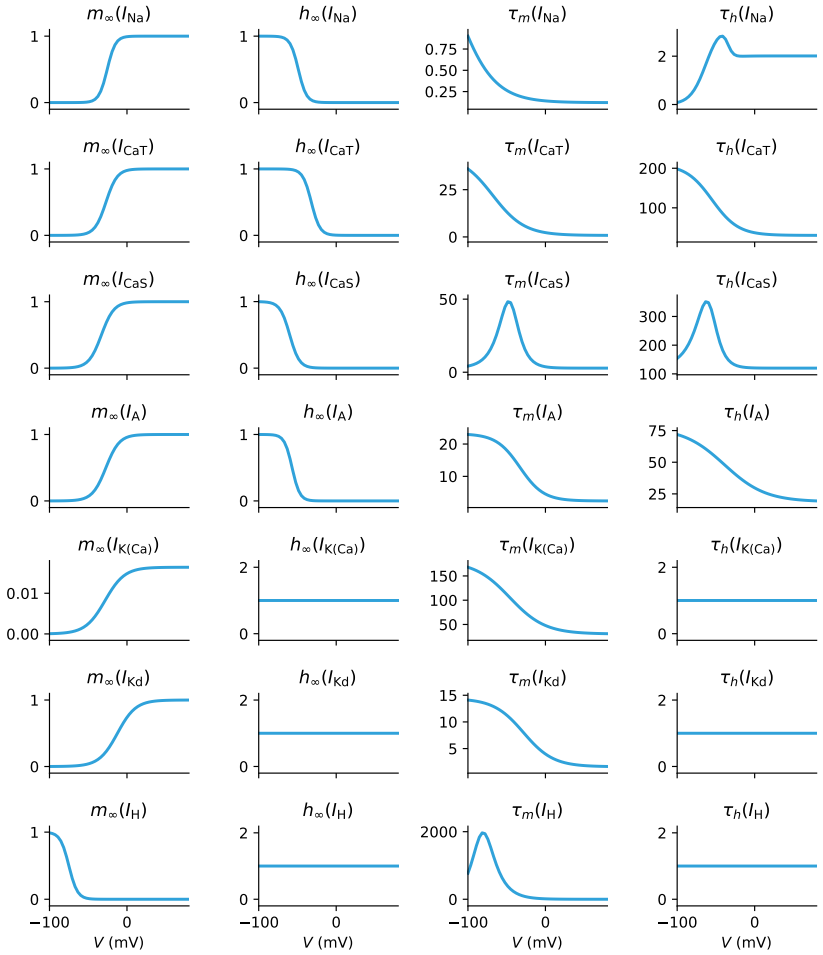


Figure A.2.: Voltage-dependent stationary values for activation variables. Note that three channel types do not have inactivating variables ($I_{K(Ca)}$, I_{Kd} and I_H). For consistency of implementation (and to allow better vectorization of Equation (A.1) (p. 156)), the corresponding h_∞ and τ_h functions are defined as constant unit functions.

A.3. Circuit model

A.3.1. Circuit topology in the original publication

This implementation does not prescribe a circuit topology, and in fact we use a simplified one with only two populations for the work in Chapter 5 (p. 99). Nevertheless, the original pyloric circuit with described in Prinz, Bucher, and Marder [21] is used as a reference model, and its parameters can be accessed with the built-in `neuron_models` variable, so it makes sense to describe it.

Recall (Section 1.2 (p. 15)) that in this model the AB and PD neurons are lumped together into a single *neuron model*,[2] with the only difference being that the AB cell has glutamatergic synapses while the PD cells have slower cholinergic synapses. This leaves us with effectively three populations, listed in Table A.7.

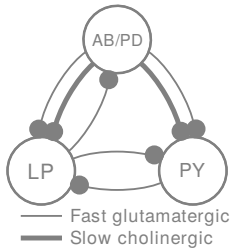


Figure A.3.: Chemical synapses of the pyloric network, reproduced from Fig. 1 of Prinz, Bucher, and Marder [21] and Figure 1.8a (p. 16) of Chapter 1 (p. 6). Electrical synapses occur only between AB and PD neurons and are not shown.

2: Recall that in Section 1.2 (p. 15) we defined the *neuron type* as one of AB, PD, LP or PY, whereas a *neuron model* refers to a particular set of conductance values chosen from Table A.4.

Table A.7.: Neuron populations of the pyloric network (reproduced from Table 1.1 (p. 16))

Population name	Cell names	Number of cells
AB/PD	anterior bursting	AB: 1
"pacemaker kernel"	pyloric dilator	PD: 2
LP	lateral pyloric	1
PY	pyloric	5–8

As can be seen in Figure A.3, nearly all populations are connected; only the PY neurons don't directly inhibit AB/PD neurons.[3]

3: Note that this is still not an "all-to-all" networks, since populations never connect to themselves.

Network connectivity is determined by the synapse strength g_s (see Equation (A.4) below). In Prinz, Bucher, and Marder [21], this parameter takes one of five values listed in Table A.8; a *circuit model* is specified by choosing one of these values for each of the edges in Figure A.3.

Table A.8.: Possible synapse values (g_s) for the pyloric network.

0 nS	1 nS (PY only)	3 nS	10 nS	30 nS	100 nS

A.3.2. Electrical synapses

The *AB* and *PD* neurons are connected via electric synapses (see Section 1.1.6.1 (p. 14)), modelled with eq. (13) from Marder and Abbott [12]:[4]

$$I_e = g_e(V_{\text{post}} - V_{\text{pre}})\,. \tag{A.3}$$

Unfortunately Prinz, Bucher, and Marder [21] do not document the value of g_e. In our implementation it is set to 1, which seems to be also what is done implicitly in their source code.

Table A.9.: Electrical synapse parameters for the pyloric circuit

g_e	1

4: As with the chemical synapses on this page, we invert the sign to match the convention in Prinz, Bucher, and Marder [21].

A.3.3. Chemical synapses

The remaining synapses in the circuit are chemical ones, and their dynamics are governed by the equations Equation (1.20) (p. 15) given in Section 1.1.6.2

(p. 14):

$$
\begin{aligned}
I_s &= g_s s(V_{\text{post}} - E_s) \\
\frac{ds}{dt} &= \frac{s_\infty(V_{\text{pre}}) - s}{\tau_s} \\
s_\infty(V_{\text{pre}}) &= \frac{1}{1 + \exp\big((V_{\text{th}} - V_{\text{pre}})/\Delta\big)} \\
\tau_s &= \frac{1 - s_\infty(V_{\text{pre}})}{k_-}
\end{aligned}
\tag{A.4}
$$

The conductivity g_s determines the synapse's strength, and therefore the network's connectivity.

The synapse parameters are given in Table A.10 and are chosen to match the inhibitory postsynaptic potentials (IPSPs) generated in the postsynaptic cell, which in turn depend on the type of neurotransmitter released by the *pre*synaptic cell. For the purposes of the implementation, this means that these values depend on the type of *pre*synaptic cell.

Table A.10.: Synaptic parameters for the pyloric circuit

Cell type	Neurotransmitter	E_s(mV)	k_-(ms^{-1})	V_{th}(mV)	Δ(mV)
pre $\in$ {*AB*, *LP*, *PY*}	glutamate	-70	1/40	-35	5
pre = *PD*	acetylcholine	-80	1/100	-35	5

In general, if we have N neurons, we would need to track a matrix of $N \times N$ synapses. However since the dynamics of s depends only on V_{pre}, it suffices to track s as a vector of length N.

A.4. Implementation

A.4.1. Some numerical considerations

Stability of the synaptic equations Equations Equation (A.4) are those reported in Prinz, Bucher, and Marder [21]. Note however that s must stay bounded between 0 and 1. Mathematically this is valid, but it requires infinitely precise integration: as s_∞ approaches 1, τ_s goes to zero and ds/dt diverges. However, such diverging derivatives will cause an adaptive numerical integrator to stall, since it must make smaller and smaller time steps in order to maintain precision. (A non-adaptive integrator will not stall, but will give incorrect results, which is arguably worse.)

In the implementation below we mitigate this issue in two ways:

1. We actually track $\tilde{s} = \log \frac{s}{1-s}$, which is the logit transform of s. This has the advantage of being unbounded, so applying a discrete update $\tilde{s}_{t+\Delta t} = \tilde{s}_t + \frac{d\tilde{s}}{dt}\Delta t$ will never produce an invalid value of $\tilde{s}$. The logit function is also monotone and thus invertible; its inverse is $s = \frac{1}{1+e^{-\tilde{s}}}$, and its derivative $\frac{d\tilde{s}}{dt} = \frac{1}{s(1-s)}\frac{ds}{dt}$.

2. We add a small $\epsilon = 10^{-8}$ to the denominator to ensure it is never exactly zero:

$$\frac{d\tilde{s}}{dt} = \frac{s_\infty(V_{\text{pre}}) - s}{\tau_s\, s(1-s) + \varepsilon}\,.$$

Memory layout of synapse variables We have two types of synapses, *glutamatergic* and *cholinergic*, and an undetermined number of neurons. To take advantage of vectorized operations, we group synapse variables according to synapse type: since parameters are then constant within each group, we can treat them as scalars, and broadcasting works with any size of voltage vector V.

The s dynamics depend only on the presynaptic neuron, so they only need to be computed once per neuron and can be tracked with a 1-d vector. This is illustrated in Table A.11, where N_c and N_g are the numbers of cholinergic and glutamatergic neurons respectively.

The synaptic currents for their part take the 2-d structure shown in Table A.12, with *row* and *column* respectively corresponding to *post*synatic and *pre*synaptic neurons:

Table A.11.: 1d memory layout for s, s_∞ and τ_∞

s_1^{chol}	…	$s_{N_c}^{\text{chol}}$	s_1^{glut}	…	$s_{N_g}^{\text{glut}}$

Table A.12.: 2-d memory layout for I_s, represented as a block matrix. Indices indicate the top left position of the block; e.g. $I^{\text{glut}}_{s,1\leftarrow N_c+1}$ is a block matrix starting at row 1 and column N_c+1.

$I^{\text{chol}}_{s,1\leftarrow 1}$	…	$I^{\text{chol}}_{s,1\leftarrow N_c}$	$I^{\text{glut}}_{s,1\leftarrow N_c+1}$	…	$I^{\text{glut}}_{s,1\leftarrow N_c+N_g}$
⋮		⋮			
$I^{\text{chol}}_{s,N_c\leftarrow 1}$	…	$I^{\text{chol}}_{s,N_c\leftarrow N_c}$	$I^{\text{glut}}_{s,N_c\leftarrow N_c+1}$	…	$I^{\text{glut}}_{s,N_c\leftarrow N_c+N_g}$
$I^{\text{chol}}_{s,N_c+1\leftarrow 1}$	…	$I^{\text{chol}}_{s,N_c+1\leftarrow N_c}$	$I^{\text{glut}}_{s,N_c+1\leftarrow N_c+1}$	…	$I^{\text{glut}}_{s,N_c+1\leftarrow N_c+N_g}$
⋮		⋮			
$I^{\text{chol}}_{s,N_c+N_g\leftarrow 1}$	…	$I^{\text{chol}}_{s,N_c+N_g\leftarrow N_c}$	$I^{\text{glut}}_{s,N_c+N_g\leftarrow N_c+1}$	…	$I^{\text{glut}}_{s,N_c+N_g\leftarrow N_c+N_g}$

Note that the implementation requires two different splittings of the set of neurons:

- by population: *AB/PD*, *LP*, and *PY*;
- by synapse type: cholinergic (*PD*) and glutamatergic (*AB/LP/PY*).

By choosing to place PD neurons first, we ensure that the synapse subsets are contiguous, which improves computational efficiency.

Initialization Virtually all dynamical systems describing natural phenomena see their the dynamic variables relax to a lower-dimensional stable manifold when integrated forward in time.[5] This is certainly true of conductance models, and we see this in Figure A.1: each of the models for PD, AB, LP and PY is initialized in some fixed generic state (see Table A.13), and after a few seconds of simulation time relaxes to its typical behaviour. It is that typical behaviour that is consistent with experiments: the initial transient behaviour represents combinations of dynamic variables that are outside the biophysical range and can therefore be ignored. The usual procedure therefore is to pick a convenient initialization state, and just integrate long enough to ensure we have reached the stable manifold. Thus we can distinguish between two types of initialization:

5: More precisely, this is true of phenomena that have some degree of robustness to perturbation: push them a little one way, and they revert back to their typical behaviour. This "typical behaviour" is the stable dimensional manifold.

Table A.13.: Cold initialization values for the pyloric model are provided on p. 4001 of Prinz, Billimoria, and Marder [15]:

V	$[Ca^{2+}]$	m	h
−50 mV	0.05 µM	0	1

Cold initialization This is a fixed initial state used for all models, when no previous simulations are available. It is used for the thermalization run, which integrates the model until it relaxes to its stable manifold.

Thermalized ("warm") initialization The final state of the initialization run is the thermalized initialization. Subsequent calls with the same model will retrieve this state from the cache.

Thermalization works because the basin of attraction[6] of the stable manifold is fairly large, so that any reasonable initial "cold" state is contained within it. As we make systems more complex however, we tend to shrink the size of a basin of attraction relative to the total state space: each new variable is one more dimension along which the system can escape the stable manifold. In particular, if we were to wire together the neurons in Figure A.1 instead of simulating them individually, they would in general *not* relax to the desired regime of sustained but bounded activity. The interaction of the non-biophysical transients at the start of Figure A.1 prevent the system as a whole from ever reaching a biophysical state.

6: Given a dynamical system on X (which maps any $(t, x) \in \mathbb{R} \times \mathcal{X}$ to $\varphi_t(x) \in \mathcal{X}$) with a stable manifold $\mathcal{S}$, the *basin of attraction* of $\mathcal{S}$ is the subset $\mathcal{X}' \subset \mathcal{X}$ such that $\lim_{t\to\infty} \varphi_t(x') \in \mathcal{S}$ for $x' \in \mathcal{X}'$.

Therefore it is essential to connect neurons together only *after* they have each individually relaxed to a reasonable state on their stable manifold. This is what we will call "*thermalizing*" the model. Typically this requires a much longer simulation time than what we need for the experiment itself, which is why we maintain a cache of thermalized states on the disk to avoid doing this every time.

During a thermalization run, the model is disconnected (all connectivities g_s and g_e are set to 0) and receives no external input; this mirrors the procedure followed by Prinz, Billimoria, and Marder [15]. Disconnecting neurons not only makes thermalization more robust, but also reduces the size of the required cache, since we only need to cache each neuron rather than each circuit.

There are five main technical differences between our procedure and that of Prinz, Bucher, and Marder [21]:

- For numerical stability, we track the values of $\operatorname{logit} m$, $\operatorname{logit} h$ and $\operatorname{logit} s$. This means we can't initialize them exactly at 0 or 1; instead we use $\operatorname{logit} m = \operatorname{logit} s = -10$ and $\operatorname{logit} h = 10$, which correspond to approximately $m = s = 10^{-5}$ and $h = 10^5$.
- Prinz, Bucher, and Marder [21] set $s = 0$ after the thermalization. This is presumably because the original neuron model catalogue did not include simulations of s. In our case, we set $s = 0$ for the cold initialization, and use its subsequent thermalized value for the data run. This is both simpler on the implementation side, and more consistent with the desire to let all transients relax before connecting neurons.
- We don't first compute the thermalization for individual model neurons separately, but instead recompute it for each combination of neuron models. (Specifically, each thermalization run is identified by a `g_cond` matrix – g_s and I_{ext} are ignored, since they are set to zero during thermalization.) If we were to simulate the entire catalog of neuron model combinations this would be wasteful, but since we only need a handful, this approach is adequate and simpler to implement.
- Instead of fixed-step Euler integration, we use the Runge-Kutta 4(5) algorithm with adaptive step sizes included in `scipy.integrate`. This is not only more efficient than Euler, but should also provide a notable improvement in numerical accuracy during sharp spikes.

- Finally, instead of detecting the steady state automatically, we use a fixed integration time and rely on visual inspection to determine whether this is enough to let transients decay in all considered models. Again we can do this because we only need to simulate a handful of models.

The main *practical* difference is that instead of pre-computing a neuron catalogue, the thermalization is done automatically and on-demand. For the user therefore it makes little difference whether a warm initialization for a particular circuit is available or not: when they request a simulation, if no warm initialization is available, they just need to wait longer to get their result. Subsequent runs then reuse the warm initialization computed on the first run.

A.4.2. Differential equation dV/dt

The evolution equations are implemented as a function `dX` for use with SciPy's ODE solvers. Since these integrate only a single vector argument, we concatenate V, $[Ca^{2+}]$, s, m and h into a single array variable X.

For efficiency reasons, `dX` implements only what is needed within the integration step. Additional operations, like memory allocation, model value retrieval, or packing/unpacking of the `X` vector, are provided as separate functions.

Note

For the purposes of presentation, the implementations given in this section have been slightly simplified. Most notably some JAX-specific implementation details were removed.

```
def dX(t, X,
       g,   # Shape: n_pops × n_channels  (3×7)  <— DOES NOT INCLUDE LEAK
       gleak,
       gs,  # Shape: n_pops × n_pops  (3×3)
       ge,  # Electrical conductance is generally fixed to 1, except
            # when we remove all connections to compute spontaneous activity
       pop_slices, # Slices for AB/PD, LP and PY
       syn_slices, # Slices for selecting by synapse type:
                   # cholinergic (PD) or glutamatergic (AB, LP, PY)
       elec_slice, # Slice for selecting the neurons with electrical synapses
       E,   # Nernst potentials
       n_neurons,  # Used for unpacking X
       I_ext = None,  # Function t → R^(#pops)
      ):
    # Shape convention: parameter × voltage
    #     (equivalent): (channel) × (neuron)

    # Unpack the vector argument
    X = X.reshape(3+nchannels+nhchannels, n_neurons)
    V, logCa, logits, logitm, logith, = (
        X[0], X[1], X[2],
        X[3:3+nchannels], X[3+nchannels:3+nchannels+nhchannels],
    )
    Ca = exp(logCa)
    s = 1 / (1 + exp(-logits))  # Inverse of the logit function
    m = 1 / (1 + exp(-logitm))  # (NB: expression remains well defined
    h = 1 / (1 + exp(-logith))  #  even when exp(logits) → ∞)


    # Update calcium reversal potentials
    E[...,[idx_ICaT,idx_ICaS],:] =  γ * (logCaout - logCa)
```

```
    # Compute conductance inputs
    mphVE  = m**p * (V-E)                          # shape      : channel × neuron
    mphVE  = mphVE.at[h_slice,:].multiply(h)
    Ii = concat(                                   # Ii.shape   : channel × neuron
        [pop_g[:,newax] * mphVE[...,slc]           # shape      : channel × pop_size
         for pop_g, slc in zip(g, pop_slices)],    # pop_g.shape: channel
        axis=-1                                    # g.shape    : pop × channel
    )

    # Compute dCa
    dCa = (-f * (Ii[...,idx_ICaT,:] + Ii[...,idx_ICaS,:]) - Ca + Ca0) / τCa
    dlogCa = dCa / Ca   # Chain rule w/ log transform

    # Compute the voltage-dependent activation variables
    m_inf, h_inf, τ_m, τ_h = act_vars(V, Ca)
    h_inf = h_inf[h_slice]  # For vectorization reasons, act_vars
    τ_h   = τ_h[h_slice]    # returns dummy values for channels with no h var

    # Compute dm and dh
    dm = (m_inf - m) / τ_m
    dh = (h_inf - h) / τ_h
    dlogitm = dm / (m * (1-m))  # Chain rule through logit
    dlogith = dh / (h * (1-h))  # Chain rule through logit

    # Compute electrical synapse inputs
    Ve = V[...,elec_slice,newax]
    Ie = ge * (Ve - swapaxes(Ve,-1,-2)).sum(axis=-1)
        # This computes Ve - Ve.T, w/ an additional time dimension on the left.
        # ge is assumed to be a scalar, currently we always either 1 or 0.
        # (0 is used to disconnect the network for thermalization.)

    # Compute the synaptic inputs
    # (By splitting glut & chol, we can use scalars for all parameters)
    cholslc, glutslc = syn_slices

    Vglut = V[...,glutslc]
    Vchol = V[...,cholslc]
    s_glut = s[...,glutslc]
    s_chol = s[...,cholslc]

    sinf_glut = 1 / (1 + exp(Vth_glut - Vglut)/Δ_glut)  # shape: (n_glut,)
    sinf_chol = 1 / (1 + exp(Vth_chol - Vchol)/Δ_chol)  # shape: (n_chol,)
    τs_glut = (1 - sinf_glut) / km_glut                 # shape: (n_glut,)
    τs_chol = (1 - sinf_chol) / km_chol                 # shape: (n_chol,)
    dlogits_glut = (sinf_glut - s_glut) / (τs_glut * s_glut * (1-s_glut) + ε)
    dlogits_chol = (sinf_chol - s_chol) / (τs_chol * s_chol * (1-s_chol) + ε)
       # Incl. chain rule w/ logit transform. ε=1e-8 is added for numerical stability

    dlogits = empty_like(logits)
    dlogits = dlogits.at[cholslc].set(dlogits_chol)
    dlogits = dlogits.at[glutslc].set(dlogits_glut)
```

```
    # Sum synaptic inputs from each population
    Is = concat(
        [sum((gs[i,j] * s[preslc] * (V[postslc,np.newaxis] - Es_tuple[j])).sum(axis=-1)
             for j, preslc in enumerate(pop_slices))
         for i, postslc in enumerate(pop_slices)],
        axis=-1
    )

    # Compute dV by summing inputs over channels
    dV = -Ii.sum(axis=-2) - Is
    # Add external inputs
    if I_ext:
        dV -= I_ext(t)
    # Add contributions currents from electrical synapses
    dV[..., elec_slice] -= Ie
    # Add leak currents
    for pop_gleak, slc in zip(gleak, pop_slices):
        dV[...,slc] -= pop_gleak * (V[...,slc] - Eleak)

    # Combine all derivatives and return
    return concat((dV.reshape(1,-1), dlogCa.reshape(1,-1),
                   dlogits.reshape(1,-1), dlogitm, dlogith)).reshape(-1)
```

A.4.3. Public API

The public API is composed of three classes: `State` on this page, `Prinz2004` (p. 169), and `SimResult` (p. 168). `Prinz2004` is the high-level interface; in many cases, this is the only class one needs to interact with directly. Internally it uses `State` and `SimResult` to translate between the human-readable dynamical variables and the flattened array optimized for numerical integration.

Thermalization interface

- The cold initialization state is given by the class method `State.cold_initialized`.
- The warm-up simulation is implemented in the method `Prinz2004.get_thermalization`.

Three class attributes of `Prinz2004` (p. 169) are used to control the behaviour of the warm-up simulation:

- `__thermalization_store__` determines where the cache is stored on disk.
- `__thermalization_time__` is the warm-up simulation time; it is currently set to 5s.
- `__thermalization_time_step__` is the recording time step for the warm-up simulation. This is only relevant for inspecting the thermalization run. (The integrator uses an adaptive time step and discards non-recorded steps.)

`State` object In order to integrate the model equations, we need to know not just the membrane potential of each neuron, but also its internal calcium concentration, synaptic variable (s) and activation and inactivation variables (m, h). These are stored together in a `State` object, which also provides

- automatic conversion to/from log-transformed values;
- convenience methods for converting between different storage layouts, for example concatening and flattening all variables for the ODE integrator, or serializing the data for storage in the thermalization cache.
- a function to construct the initial “cold” initialization state.

```python
@dataclass
class State:
  V     : Array[float, 1]  # (n_neurons)
  logCa : Array[float, 1]  # (n_neurons)
  logits: Array[float, 1]  # (n_neurons)
  logitm: Array[float, 2]  # (n_neurons, n_channels)
  logith: Array[float, 2]  # (n_neurons, n_channels)

  """Storage container for simulator variables.

  - Handles automatic conversion to/from log-transformed values.
  - Provides methods for converting between storage layouts, in particular for
  exporting to the 1-d vector format required by ODE integrators.
  """

  @classmethod
  def cold_initialized(cls, n_neurons):
    """Default state initialization. Used to initialize the thermalization run."""
    return cls(
      -50   * np.ones(n_neurons),
      np.log(0.05) * np.ones(n_neurons),
      -10 * np.ones(n_neurons),                    # "fully deactivated"
      -10 * np.ones((n_neurons, len(channels))),  # "fully deactivated"
      10 * np.ones((n_neurons, nhchannels)),      # "fully activated"
    )

  @property
  def s(self):
    return 1 / (1 + 1/np.exp(self.logits))  # Inverse of the logit function
      # (NB: expression chosen to be well defined even when exp(logits) → ∞)
  @property
  def m(self):
    return 1 / (1 + 1/np.exp(self.logitm))
  @property
  def h(self):
    return 1 / (1 + 1/np.exp(self.logith))
  @property
  def Ca(self):
    return np.exp(self.logCa)
```

`SimResult` object The ODE integrators provided by `scipy.integrate` treat the data as a flat 1-D vector, which is not convenient for associating the trace of each component to the correct state variable. A `SimResult` stores the data returned from the integrator, in the integrator's compact format, but provides a human-friendly interface for retrieving traces for individual variables. Conversions to/from log-transformed values are handled automatically.

```
@dataclass
class SimResult:
  t         : Array[float, 1]
  data      : Array[float, 3]
  pop_slices: List[slice]

  """Stores and makes accessible the results of a simulation run.

  Underlying storage is very close to the output of the ODE simulator.
  Properties for each variable (`V`, `logCa`, `Ca`, etc.) retrieve the correct
  rows from the data structure.
  """

  def __post_init__(self):
    self.data = self.data.reshape(3+len(channels)+nhchannels, -1, len(self.t))
    self.data = np.moveaxis(self.data, -1, 0)
       # Both Pandas & Holoviews work better if time is the index axis

  def __getitem__(self, key) -> Union[SimResult,State]:
    """Indexing by time returns either a `State` (single t)
    or a `SimResult` with a reduced time window (multiple t)
    """
    if isinstance(key, (slice, list, np.ndarray)):
      return SimResult(t=self.t[key], data=self.data[key,:,:],
                       pop_slices=self.pop_slices)
    elif isinstance(key, int):
      return self.t, State.from_array(self.data[key,:,:])
    else:
      raise TypeError("SimResult` only supports 1-d indexing along the time axis.")
```

```
  @property
  def V(self): return self._make_df(self.data[:, 0, :])
  @property
  def logCa(self): return self._make_df(self.data[:, 1, :])
  @property
  def Ca(self): return np.exp(self.logCa)
  @property
  def logits(self): return self._make_df(self.data[:, 2, :])
  @property
  def s(self): return 1 / (1 + np.exp(-self.logits))
  @property
  def logitm(self): return self.data[:, 3:3+len(channels), :]
  @property
  def m(self): return 1 / (1 + np.exp(-self.logitm))
  @property
  def logith(self): return self.data[:, 3+len(channels):3+len(channels)+nhchannels, :]
  @property
  def h(self): return 1 / (1 + np.exp(-self.logith))

  def _make_df(self, data) -> pd.DataFrame:
    """Return the traces as a Pandas DataFrame."""
    cols = pd.MultiIndex.from_tuples(
            ((pop, i)
             for pop, slc in self.pop_slices.items()
             for i in range(1, 1+slc.stop-slc.start)),
            names=["pop", "index"]
      )
    return pd.DataFrame(data, index=pd.Index(self.t, name="time"), columns=cols)
```

`Prinz2004` object This is the core simulator class. To perform a simulation, create an instance of this class as in A.1 and call it as in A.2.

```
@dataclass(frozen=True)
class Prinz2004:
  pop_sizes  : dict[str, int]
  gs         : Array[float, 2]
  g_ion: Optional[Array[float, 2]]=None
  ge         : float=1.  # Currently only used to turn on/off electrical connectivity

  """Core simulator class.
  To perform a simulation, create an instance of this class and call it
  with an array time points to be recorded.
  """

  # Private attributes
  __thermalization_store__ : ClassVar[Path] = \
      config.paths.simresults/"prinz2004_thermalize"
  __thermalization_time__  : ClassVar[float] = 5000.
  __thermalization_time_step__: ClassVar[float] = 1.
  _thermalization_store_lock: ClassVar[Lock] = \
      Lock(str(__thermalization_store__.with_suffix(".lock")))
  _thermalization_store_lock.lifetime = timedelta(seconds=15)  # 15s is the default
```

```
## Public API

def __call__(self, t_array, I_ext: Optional[Callable]=None):
  X0 = self.get_thermalization()
  res = self.integrate(0, X0, t_array, I_ext)
  return SimResult(t_array, res.y, self.pop_slices)

def derivative(self, t: float, X: State, I_ext: Optional[Callable]=None):
  """
  Evaluate the model equations, returning the derivative at `t` if the state is `X`.
  This is a convenience method, equivalent to the code in `integrate`, but distinct.
  """
  X0 = X
  # C..\@ self.integrate()
  if isinstance(X0, State):
    X0_flat = X0.to_vector()
  else:
    X0_flat = X0.flatten()
    X0 = self.State.from_vector(X0)
  args=(self.g, self.gleak, self.gs, self.ge,  # Conductances
        self.pop_slices, self.syn_slices, self.elec_slice, # Population slices
        self.E(X0.Ca),                       # Expanded from constants.E
        self.tot_cells,                      # Pre-computed number of neurons
        I_ext)                               # External input, if provided
  return dX(t, X0_flat, *args)

## Semi-private methods

@classmethod
def clear_thermalization_store(cls):
  cls.__thermalization_store__.unlink(missing_ok=True)

def integrate(self, t0: float, X0: Union[Array, State], t_eval: ArrayLike,
              I_ext: Optional[Callable]=None) -> OdeResult:
  """
  Integrate the model from the initial state `(t0, X0)`.
  The result will contain the states at all values in `t_eval`.
  Args:
      t0: time corresponding to X0
      X0: Initial state; either an instance of `State` or a flat vector.
      t_eval: The time points at which to record the trace
      I_ext: If provided, should be a function with signature ``(t) → I``,
         where ``I`` is a 1-d vector with one element for every neuron.
         The units of I are nA; its value is added directly to the derivative dV.
         Alternatively, if all neurons receive the same input, ``I`` can be a scalar.
  """
  if isinstance(X0, State):
    X0_flat = X0.to_vector()
  else:
    X0_flat = X0.flatten()
    X0 = self.State.from_vector(X0)
```

```
  t_eval = np.sort(t_eval)
  T = t_eval[-1]

  res = integrate.solve_ivp(
    dX, (t0, T), X0_flat, method="RK45", t_eval=t_eval,
    args=(self.g, self.gleak, self.gs, self.ge,  # Conductances
          self.pop_slices, self.syn_slices, self.elec_slice,  # Population slices
          self.E(X0.Ca),
          self.tot_cells,                       # Pre-computed number of neurons
          I_ext),                               # External input, if provided
    first_step=0.005)

  if res.status < 0:
    raise RuntimeError(f"Integration failed with the message:\n{res.message}")
  return res

@property
def g_cond(self):
  # Explicitely assigned g_cond
  return pd.DataFrame(self.g_ion,
                      index=self.pop_model_list, columns=g_cond.columns)
@property
def g(self):
  return self.g_cond.filter(regex=r"mathrm{(?!leak)").to_numpy()
@property
def gleak(self):
  return self.g_cond.loc[:, '$g(I_\\mathrm{leak})$'].to_numpy()

def E(self, Ca):
  E = np.tile(constants.E[:,np.newaxis], (1, self.tot_cells))
  E = E.at[[1,2],:].set(constants.y*np.log(constants.Caout/Ca))
  return E

def get_thermalization(self) -> State:
  """
  The thermalization is obtained by integrating the network model
  with all connections to zero, to find the steady-state of each neuron model.
  This method either performs that integration, or, if it was already
  done before, retrieves it from a cache.
  """
  sim = self.thermalize()
  t, X = sim[-1]  # Get the final state
  # Warm init is done with one neuron per population: reinflate the populations.
  # NB: The thermalization uses the merged AB/PD populations of pop_slices
  pop_sizes = tuple(slc.stop-slc.start for slc in self.pop_slices.values())
  return State.from_array(np.repeat(X.to_array(), pop_sizes, axis=-1))

def thermalize(self) -> SimResult:
  """
  Compute the initial simulation used to thermalize the model.
  - Constructs a surrogate model with all population sizes set to 1.
  - Sets all connectivities to 0.
  - Integrate
```

```
    In addition, this method manages the termalization cache, and will skip
    all steps if it finds a result already in the cache.

    Returns a `SimResult` instance, with recording resolution determined by
    ``self.__thermalization_time_step__``.

    Normally this method is not used directly, but called by `get_thermalization`.
    The main reason to use it directly would be to plot the initialization
    run for inspection.
    """
    # We thermalize the model by turning off synaptic connections,
    # so we also use a key which only depends on the model identities.
    init_key = str( (("T", self.__thermalization_time__),
                     ("g_cond", self.g_cond.to_csv()))   )
    if not self.__thermalization_store__.exists():
      self.__thermalization_store__.parent.mkdir(parents=True, exist_ok=True)
      warm_t_X = None
    else:
      with shelve.open(str(self.__thermalization_store__), 'r') as store:
        warm_t_X = store.get(init_key)
    if warm_t_X:  # We found a cached thermalization result => use it
      t, warm_X = warm_t_X
    else:         # We did not find a cached thermalization result
      # Get the cold initialized state (for a smaller model with 1 neuron / pop)
      X0 = State.cold_initialized(len(self.pop_slices))
      # Construct the array of times at which we will record
      init_T = self.__thermalization_time__
      Δt = self.__thermalization_time_step__
      t_eval = np.concatenate((np.arange(0, init_T, Δt), [init_T]))
      # Build the model with no connections and only one neuron per pop
      disconnected_model = Prinz2004(
          pop_sizes={pop: 1 for pop in self.pop_slices},
          gs=np.zeros_like(self.gs),
          g_ion=self.g_ion,
          ge=0)
      # Integrate
      res = disconnected_model.integrate(0, X0, t_eval)
      t, warm_X = res.t, res.y
      # Update the cache
      with self._thermalization_store_lock:  # lock b/c shelve is not thread safe
          with shelve.open(str(self.__thermalization_store__)) as store:
              store[init_key] = (t, warm_X)
    # Return
    return SimResult(t, warm_X,
                     pop_slices={pop: slice(i,i+1)
                                 for i,pop in enumerate(self.pop_slices)})
```

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cited on pages 2, 38, 41, 64).

[2] Salvatore Cuomo et al. "Scientific Machine Learning Through Physics–Informed Neural Networks: Where We Are and What's Next". In: *Journal of Scientific Computing* 92.3 (July 26, 2022), p. 88. DOI: 10.1007/s10915-022-01939-z (cited on page 3).

[3] Casey Henley. *Foundations of Neuroscience*. Michigan State University Libraries, Jan. 1, 2021 (cited on pages 6, 14).

[4] Hans G. L. Coster. "Chapter 2 - Dielectric and Electrical Properties of Lipid Bilayers in Relation to Their Structure." In: *Planar Lipid Bilayers (BLMs) and Their Applications*. Ed. by H. T. Tien and A. Ottova-Leitmannova. Membrane Science and Technology 7. Amsterdam: Elsevier Science B.V., Jan. 1, 2003, pp. 75–108. DOI: 10.1016/S0927-5193(03)80026-8 (cited on page 7).

[5] John W. Moore and Ann E. Stuart. *The Bilayer Membrane Is a Capacitor*. 2018. URL: https://www.sas.upenn.edu/LabManuals/BBB251/NIA/NEUROLAB/aframes.htm (visited on 08/09/2025) (cited on page 7).

[6] R. K. Pathria and Paul D. Beale. *Statistical Mechanics*. 3rd ed. Amsterdam ; Boston: Elsevier/Academic Press, 2011. 718 pp. (cited on page 8).

[7] Eugene M. Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. 1. MIT Press paperback ed. Computational Neuroscience 21. Cambridge, Mass.: MIT Press, Aug. 2007. 441 pp. (cited on pages 9, 10, 12).

[8] Wulfram Gerstner et al. *Neuronal Dynamics from Single Neurons to Networks and Models of Cognition*. Cambridge: Cambridge University Press, 2014 (cited on pages 9, 10, 12–15, 21, 23, 24, 69, 78).

[9] Bertil Hille. *Ion Channels of Excitable Membranes*. 3. ed. Sunderland, Mass: Sinauer Assoc, 2001. 814 pp. (cited on pages 9, 10, 13).

[10] Sang-Jun Park et al. "OPRLM: Orientations of Proteins in Realistic Lipid Membranes". In: *Biophysical Journal* 122.3 (Feb. 10, 2023), 283a. DOI: 10.1016/j.bpj.2022.11.1608 (cited on page 10).

[11] J. Gordon Betts et al. "The Cell Membrane". In: *Anatomy and Physiology*. Houston, Texas: OpenStax, Apr. 25, 2013 (cited on page 11).

[12] Eve Marder and L F Abbott. "Modeling Small Networks". In: *Methods in Neuronal Modeling: From Ions to Networks*. Ed. by Christof Koch and Idan Segev. 2nd ed. Computational Neuroscience. Cambridge, Mass: MIT Press, 1998 (cited on pages 12, 14, 15, 153, 156, 160).

[13] Peter Dayan and L. F. Abbott. *Theoretical Neuroscience : Computational and Mathematical Modeling of Neural Systems*. Computational Neuroscience. Cambridge, Mass: Massachusetts Institute of Technology Press, 2001 (cited on pages 12, 14).

[14] A. L. Hodgkin and A. F. Huxley. "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve". In: *The Journal of Physiology* 117.4 (Aug. 28, 1952), pp. 500–544 (cited on page 12).

[15] Astrid A. Prinz, Cyrus P. Billimoria, and Eve Marder. "Alternative to Hand-Tuning Conductance-Based Models: Construction and Analysis of Databases of Model Neurons". In: *Journal of Neurophysiology* 90.6 (Dec. 2003), pp. 3998–4015. DOI: 10.1152/jn.00641.2003 (cited on pages 12, 14–17, 123, 127, 153, 156, 157, 162, 163).

[16] Christof Koch and Idan Segev. *Methods in Neuronal Modeling: From Ions to Networks*. 2nd ed. Cambridge, Mass.: MIT Press, 1998 (cited on page 13).

[17] Benjamin Barlow. "Simulating Action Potential Initiation and Propagation in Physically Detailed Damaged and Healthy Neurons". Université d'Ottawa / University of Ottawa, 2025 (cited on page 13).

[18] Louis Jacques. "Action Potential Initiation in a Damaged Axon Initial Segment". Université d'Ottawa / University of Ottawa, 2016 (cited on page 13).

[19] Kevin Fu-Hsiang Lee. "Dynamics of Synapse Function during Postnatal Development and Homeostatic Plasticity in Central Neurons". Thesis Ph.D.–University of Ottawa., 2015 (cited on page 13).

[20] Tilo Schwalger, Moritz Deger, and Wulfram Gerstner. "Towards a Theory of Cortical Columns: From Spiking Neurons to Interacting Neural Populations of Finite Size". In: *PLOS Computational Biology* 13.4 (Apr. 19, 2017), e1005507. DOI: 10.1371/journal.pcbi.1005507 (cited on pages 14, 24, 26, 27, 64–68, 72, 73, 76, 78, 80, 86, 91, 92).

[21] Astrid A Prinz, Dirk Bucher, and Eve Marder. "Similar Network Activity from Disparate Circuit Parameters". In: *Nature Neuroscience* 7.12 (Dec. 2004), pp. 1345–1352. DOI: 10.1038/nn1352 (cited on pages 15, 16, 99–101, 104, 105, 123, 126–128, 153, 154, 156, 157, 160, 161, 163).

[22] Leandro M Alonso and Eve Marder. "Temperature Compensation in a Small Rhythmic Circuit". In: *eLife* 9 (June 2, 2020). Ed. by Frances K Skinner et al., e55470. DOI: 10.7554/eLife.55470 (cited on pages 15, 19).

[23] Tilman Kispersky, Gabrielle J. Gutierrez, and Eve Marder. "Functional Connectivity in a Rhythmic Inhibitory Circuit Using Granger Causality". In: *Neural Systems & Circuits* 1.1 (May 25, 2011), p. 9. DOI: 10.1186/2042-1001-1-9 (cited on pages 15, 16).

[24] Pedro J Gonçalves et al. "Training Deep Neural Density Estimators to Identify Mechanistic Models of Neural Dynamics". In: *eLife* 9 (Sept. 17, 2020). Ed. by John R Huguenard, Timothy O'Leary, and Mark S Goldman, e56261. DOI: 10.7554/eLife.56261 (cited on pages 19, 20, 100, 105, 116, 153).

[25] Richard B. Stein. "A Theoretical Analysis of Neuronal Variability". In: *Biophysical Journal* 5.2 (Mar. 1, 1965), pp. 173–194. DOI: 10.1016/S0006-3495(65)86709-1 (cited on page 21).

[26] Bruce W. Knight. "Dynamics of Encoding in a Population of Neurons". In: *Journal of General Physiology* 59.6 (June 1, 1972), pp. 734–766. DOI: 10.1085/jgp.59.6.734 (cited on page 21).

[27] Nicolas Brunel and Mark C. W. van Rossum. "Lapicque's 1907 Paper: From Frogs to Integrate-and-Fire". In: *Biological Cybernetics* 97.5 (Dec. 1, 2007), pp. 337–339. DOI: 10.1007/s00422-007-0190-0 (cited on page 21).

[28] Gábor Molnár et al. *Human Pyramidal to Interneuron Synapses Are Mediated by Multi-Vesicular Release and Multiple Docked Vesicles*. eLife. Aug. 18, 2016. DOI: 10.7554/eLife.18167 (cited on page 23).

[29] Eric R. Kandel. "Dale's Principle and the Functional Specificity of Neurons[1]". In: *Psychopharmacology, a Review of Progress, 1957-1967*. Sixth Annual Meeting of the American College of Neuropsychopharmacology. San Juan, Puerto Rico, 1957, pp. 385–398 (cited on page 23).

[30] Jonathan Cornford et al. "Learning to Live with Dale's Principle: ANNs with Separate Excitatory and Inhibitory Units". In: International Conference on Learning Representations. Oct. 2, 2020 (cited on page 23).

[31] Adam Haber and Elad Schneidman. "The Computational and Learning Benefits of Daleian Neural Networks". In: *Advances in Neural Information Processing Systems* 35 (Dec. 6, 2022), pp. 5194–5206 (cited on page 23).

[32] Pingsheng Li et al. "Learning Better with Dale's Law: A Spectral Perspective". In: *Advances in Neural Information Processing Systems* 36 (Dec. 15, 2023), pp. 944–956 (cited on page 23).

[33] Aishwarya H. Balwani et al. "Constructing Biologically Constrained RNNs via Dale's Backprop and Topologically-Informed Pruning". In: *bioRxiv* (Jan. 13, 2025), p. 2025.01.09.632231. DOI: 10.1101/2025.01.09.632231 (cited on page 23).

[34] Naoshige Uchida. "Bilingual Neurons Release Glutamate and GABA". In: *Nature Neuroscience* 17.11 (Nov. 2014), pp. 1432–1434. DOI: 10.1038/nn.3840 (cited on page 23).

[35] Peter Jonas, Josef Bischofberger, and Jürgen Sandkühler. "Corelease of Two Fast Neurotransmitters at a Central Synapse". In: *Science* 281.5375 (July 17, 1998), pp. 419–424. DOI: 10.1126/science.281.5375.419 (cited on page 23).

[36] Erik Svensson et al. "General Principles of Neuronal Co-transmission: Insights From Multiple Model Systems". In: *Frontiers in Neural Circuits* 12 (Jan. 21, 2019). DOI: 10.3389/fncir.2018.00117 (cited on page 23).

[37] Wulfram Gerstner and Richard Naud. "How Good Are Neuron Models?" In: *Science* 326.5951 (Oct. 16, 2009), pp. 379–380. DOI: 10.1126/science.1181936 (cited on page 24).

[38] S. Mensi et al. "Parameter Extraction and Classification of Three Cortical Neuron Types Reveals Two Distinct Adaptation Mechanisms". In: *Journal of Neurophysiology* 107.6 (Mar. 15, 2012), pp. 1756–1775. DOI: 10.1152/jn.00408.2011 (cited on pages 24, 63).

[39] D. R. Cox. *Renewal Theory*. Methuen, 1962. 166 pp. (cited on page 25).

[40] Wulfram Gerstner. "Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking". In: *Neural Computation* 12.1 (Jan. 1, 2000), pp. 43–89. DOI: 10.1162/089976600300015899 (cited on pages 25, 64, 78).

[41] Richard Naud and Wulfram Gerstner. "Coding and Decoding with Adapting Neurons: A Population Approach to the Peri-Stimulus Time Histogram". In: *PLOS Computational Biology* 8.10 (Oct. 4, 2012), e1002711. DOI: 10.1371/journal.pcbi.1002711 (cited on pages 25, 26, 67, 92).

[42] Tobias C. Potjans and Markus Diesmann. "The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model". In: *Cerebral Cortex* 24.3 (Mar. 2014), pp. 785–806. DOI: 10.1093/cercor/bhs358 (cited on pages 27, 63, 65, 72).

[43] Tilo Schwalger. "Mapping Input Noise to Escape Noise in Integrate-and-Fire Neurons: A Level-Crossing Approach". In: *Biological Cybernetics* 115.5 (Oct. 1, 2021), pp. 539–562. DOI: 10.1007/s00422-021-00899-1 (cited on page 27).

[44] Grégory Dumont, Alexandre Payeur, and André Longtin. "A Stochastic-Field Description of Finite-Size Spiking Neural Networks". In: *PLOS Computational Biology* 13.8 (Aug. 7, 2017), e1005691. DOI: 10.1371/journal.pcbi.1005691 (cited on pages 27, 64).

[45] Tilo Schwalger and Anton V Chizhov. "Mind the Last Spike — Firing Rate Models for Mesoscopic Populations of Spiking Neurons". In: *Current Opinion in Neurobiology* 58 (Oct. 1, 2019), pp. 155–166. DOI: 10.1016/j.conb.2019.08.003 (cited on pages 27, 78).

[46] Kevin P Murphy. *Machine Learning a Probabilistic Perspective*. Cambridge, Mass.: MIT Press, 2012 (cited on pages 31, 35, 38, 40, 41).

[47] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519 (cited on page 31).

[48] Vladimir N Vapnik. *The Nature of Statistical Learning Theory*. New York, NY: Springer New York, 2000 (cited on pages 32, 35, 54, 55, 103).

[49] Jose M. Bernardo. "Expected Information as Expected Utility". In: *The Annals of Statistics* 7.3 (May 1, 1979). DOI: 10.1214/aos/1176344689 (cited on page 34).

[50] Tilmann Gneiting and Adrian E Raftery. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477 (Mar. 2007), pp. 359–378. DOI: 10.1198/016214506000001437 (cited on pages 34, 103).

[51] JoramSoch. "Proof: Gibbs' Inequality". In: *The Book of Statistical Proofs*. Sept. 9, 2020, Proof #164. DOI: 10.5281/zenodo.4305949 (cited on page 34).

[52] David Williams. *Weighing the Odds: A Course in Probability and Statistics*. 1st ed. West Nyack: Cambridge University Press, 2001. 1 p. (cited on pages 34, 59).

[53] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer, 2006. 738 pp. (cited on pages 34, 35, 38).

[54] Matthew Parry, A. Philip Dawid, and Steffen Lauritzen. "Proper Local Scoring Rules". In: *The Annals of Statistics* 40.1 (Feb. 2012), pp. 561–592. DOI: 10.1214/12-AOS971 (cited on pages 34, 103).

[55] Sumio Watanabe. "Equations of States in Singular Statistical Estimation". In: *Neural Networks* 23.1 (Jan. 1, 2010), pp. 20–34. DOI: 10.1016/j.neunet.2009.08.002 (cited on pages 34, 51).

[56] Aki Vehtari, Andrew Gelman, and Jonah Gabry. "Practical Bayesian Model Evaluation Using Leave-One-out Cross-Validation and WAIC". In: *Statistics and Computing* 27.5 (Sept. 1, 2017), pp. 1413–1432. DOI: 10.1007/s11222-016-9696-4 (cited on pages 34, 50, 51, 104, 121, 126, 141).

[57] David Swigon et al. "On the Importance of the Jacobian Determinant in Parameter Inference for Random Parameter and Random Measurement Error Models". In: *SIAM/ASA Journal on Uncertainty Quantification* 7.3 (Jan. 2019), pp. 975–1006. DOI: 10.1137/17M1114405 (cited on pages 36, 38, 124).

[58] Andrew Gelman, Daniel Simpson, and Michael Betancourt. "The Prior Can Often Only Be Understood in the Context of the Likelihood". In: *Entropy* 19.10 (2017). DOI: 10.3390/e19100555 (cited on page 36).

[59] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge, UK ; New York: Cambridge University Press, 2003. 628 pp. (cited on pages 37, 60, 119, 121).

[60] Michael Betancourt. "A Conceptual Introduction to Hamiltonian Monte Carlo". Jan. 9, 2017 (cited on page 37).

[61] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. "Probabilistic Programming in Python Using PyMC3". In: *PeerJ Computer Science* 2 (Apr. 6, 2016), e55. DOI: 10.7717/peerj-cs.55 (cited on pages 37, 81, 82).

[62] Radford M. Neal. "An Improved Acceptance Procedure for the Hybrid Monte Carlo Algorithm". In: *Journal of Computational Physics* 111.1 (Mar. 1, 1994), pp. 194–203. DOI: 10.1006/jcph.1994.1054 (cited on page 37).

[63] Radford M. Neal. "MCMC Using Hamiltonian Dynamics". June 8, 2012 (cited on pages 37, 72, 81).

[64] Alexandre René. "Spectral Solution Method for Distributed Delay Stochastic Differential Equations". MA thesis. Ottawa, Canada: Université d'Ottawa / University of Ottawa, 2016. 95 pp. (cited on page 38).

[65] Patrick Kidger and Terry Lyons. "Universal Approximation with Deep Narrow Networks". In: *Proceedings of Thirty Third Conference on Learning Theory*. Conference on Learning Theory. PMLR, July 15, 2020, pp. 2306–2327 (cited on page 39).

[66] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals, and Systems* 2.4 (Dec. 1989), pp. 303–314. DOI: 10.1007/BF02551274 (cited on page 39).

[67] Yann A. LeCun et al. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer, 2012, pp. 9–48. DOI: 10.1007/978-3-642-35289-8_3 (cited on page 40).

[68] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". Dec. 22, 2014 (cited on pages 40, 80).

[69] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0 (cited on page 41).

[70] Léon Bottou. "Stochastic Gradient Descent Tricks". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 421–436. DOI: 10.1007/978-3-642-35289-8_25 (cited on page 42).

[71] Theano Development Team et al. "Theano: A Python Framework for Fast Computation of Mathematical Expressions". May 9, 2016 (cited on page 43).

[72] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. Fourth edition. Chicago ; London: The University of Chicago Press, 2012. 217 pp. (cited on page 44).

[73] Sumio Watanabe. "Almost All Learning Machines Are Singular". In: *2007 IEEE Symposium on Foundations of Computational Intelligence*. 2007 IEEE Symposium on Foundations of Computational Intelligence. Apr. 2007, pp. 383–388. DOI: 10.1109/FOCI.2007.371500 (cited on page 48).

[74] R. M. Golden. "Discrepancy Risk Model Selection Test Theory for Comparing Possibly Misspecified or Nonnested Models". In: *Psychometrika* 68.2 (June 2003), pp. 229–249. DOI: 10.1007/BF02294799 (cited on pages 48, 100, 108, 124).

[75] V. Vapnik. "Principles of Risk Minimization for Learning Theory". In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson, and R.P. Lippmann. Vol. 4. Morgan-Kaufmann, 1991 (cited on pages 49, 100, 103).

[76] Andrew Gelman et al. "Bayesian Data Analysis (Online)". Textbook. Online edition Bayesian Data Analysis 3rd edition (CRC Press), with errors fixed as of 15 February 2021. 2021 (cited on pages 50, 54, 56).

[77] Sumio Watanabe. "Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory". In: *Journal of Machine Learning Research* 11.116 (2010), pp. 3571–3594 (cited on page 50).

[78] Ravin Kumar et al. "ArviZ a Unified Library for Exploratory Analysis of Bayesian Models in Python". In: *Journal of Open Source Software* 4.33 (Jan. 15, 2019), p. 1143. DOI: 10.21105/joss.01143 (cited on pages 51, 121).

[79] Andrew Gelman and Yuling Yao. "Holes in Bayesian Statistics". Version 2. In: *Journal of Physics G: Nuclear and Particle Physics* 48.1 (Jan. 1, 2021), p. 014002. DOI: 10.1088/1361-6471/abc3a5 (cited on pages 52–54, 100, 121).

[80] John Skilling. "Nested Sampling for General Bayesian Computation". In: *Bayesian analysis* 1.4 (2006), pp. 833–859 (cited on pages 52, 75, 121).

[81] Joshua S. Speagle. "Dynesty: A Dynamic Nested Sampling Package for Estimating Bayesian Posteriors and Evidences". Apr. 3, 2019 (cited on page 52).

[82] Robert E. Kass and Adrian E. Raftery. "Bayes Factors". In: *Journal of the American Statistical Association* 90.430 (1995), pp. 773–795. DOI: 10.2307/2291091 (cited on pages 52, 53).

[83] Marco Taboga. "Jeffreys' Scale". In: *Lectures on Probability Theory and Mathematical Statistics*. Kindle Direct Publishing, 2021, Online appendix (cited on page 52).

[84] Roberto Trotta. "Bayes in the Sky: Bayesian Inference and Model Selection in Cosmology". In: *Contemporary Physics* 49.2 (Mar. 1, 2008), pp. 71–104. DOI: 10.1080/00107510802066753 (cited on pages 52, 119, 121, 141).

[85] Harold Jeffreys. "Some Tests of Significance, Treated by the Theory of Probability". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 31.2 (Apr. 1935), pp. 203–222. DOI: 10.1017/S030500410001330X (cited on page 52).

[86] Gideon Schwarz. "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2 (1978), pp. 461–464 (cited on pages 53, 54, 106, 121).

[87] David J. Spiegelhalter et al. "The Deviance Information Criterion: 12 Years On". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 76.3 (June 1, 2014), pp. 485–493. DOI: 10.1111/rssb.12062 (cited on pages 54, 106).

[88] David F. Findley. "Counterexamples to Parsimony and BIC". In: *Annals of the Institute of Statistical Mathematics* 43.3 (Sept. 1, 1991), pp. 505–514. DOI: 10.1007/BF00053369 (cited on pages 54, 103).

[89] J. Rissanen. "Modeling by Shortest Data Description". In: *Automatica* 14.5 (Sept. 1, 1978), pp. 465–471. DOI: 10.1016/0005-1098(78)90005-5 (cited on page 54).

[90] A. Barron, J. Rissanen, and Bin Yu. "The Minimum Description Length Principle in Coding and Modeling". In: *IEEE Transactions on Information Theory* 44.6 (Oct. 1998), pp. 2743–2760. DOI: 10.1109/18.720554 (cited on page 54).

[91] Peter D. Grünwald. *The Minimum Description Length Principle*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2007. 1 p. (cited on page 54).

[92] Peter Grünwald and Teemu Roos. "Minimum Description Length Revisited". In: *International Journal of Mathematics for Industry* 11.1 (Mar. 12, 2020), p. 1930001. DOI: 10.1142/S2661335219300018 (cited on pages 54–56, 121).

[93] Peter Grünwald and Thijs van Ommen. "Inconsistency of Bayesian Inference for Misspecified Linear Models, and a Proposal for Repairing It". In: *Bayesian Analysis* 12.4 (Dec. 2017), pp. 1069–1103. DOI: 10.1214/17-BA1085 (cited on pages 56, 58, 103).

[94] Samuel Kotz and Norman L. Johnson, eds. *Breakthroughs in Statistics: Foundations and Basic Theory*. Vol. 1. 3 vols. Springer Series in Statistics, Perspectives in Statistics. New York, NY: Springer, 1992. 632 pp. DOI: 10.1007/978-1-4612-0919-5 (cited on page 60).

[95] Chandelier. *Answer to "Comparing Non Nested Models with AIC"*. Cross Validated. Sept. 26, 2014. URL: https://stats.stackexchange.com/a/116943 (visited on 02/05/2025) (cited on page 60).

[96] Alexandre René, André Longtin, and Jakob H. Macke. "Inference of a Mesoscopic Population Model from Population Spike Trains". In: *Neural Computation* 32.8 (June 10, 2020), pp. 1448–1498. DOI: 10.1162/neco_a_01292 (cited on pages 63, 99, 100, 123).

[97] Jonathan W. Pillow et al. "Spatio-Temporal Correlations and Visual Signalling in a Complete Neuronal Population". In: *Nature* 454.7207 (Aug. 2008), pp. 995–999. DOI: 10.1038/nature07140 (cited on pages 63, 74).

[98] Jakob H Macke et al. "Empirical Models of Spiking in Neural Populations". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 1350–1358 (cited on pages 63, 74, 75).

[99] Yuan Zhao and Il Memming Park. "Interpretable Nonlinear Dynamic Modeling of Neural Trajectories". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 3333–3341 (cited on pages 63, 74).

[100] Chethan Pandarinath et al. "Inferring Single-Trial Neural Population Dynamics Using Sequential Auto-Encoders". In: *Nature Methods* 15.10 (Oct. 2018), p. 805. DOI: 10.1038/s41592-018-0109-9 (cited on page 63).

[101] Michael Hawrylycz et al. "Inferring Cortical Function in the Mouse Visual System through Large-Scale Systems Neuroscience". In: *Proceedings of the National Academy of Sciences* 113.27 (July 5, 2016), pp. 7337–7344. DOI: 10.1073/pnas.1512901113 (cited on page 63).

[102] Brent Doiron et al. "The Mechanics of State-Dependent Neural Correlations". In: *Nature Neuroscience* 19.3 (Mar. 2016), pp. 383–393. DOI: 10.1038/nn.4242 (cited on page 63).

[103] Daniel Martí, Nicolas Brunel, and Srdjan Ostojic. "Correlations between Synapses in Pairs of Neurons Slow down Dynamics in Randomly Connected Neural Networks". In: *Physical Review E* 97.6 (June 26, 2018), p. 062314. DOI: 10.1103/PhysRevE.97.062314 (cited on page 63).

[104] Edward Wallace et al. "Emergent Oscillations in Networks of Stochastic Spiking Neurons". In: *PLOS ONE* 6.5 (May 6, 2011), e14804. DOI: 10.1371/journal.pone.0014804 (cited on page 64).

[105] Duane Q. Nykamp and Daniel Tranchina. "A Population Density Approach That Facilitates Large-Scale Modeling of Neural Networks: Analysis and an Application to Orientation Tuning". In: *Journal of Computational Neuroscience* 8.1 (Jan. 1, 2000), pp. 19–50. DOI: 10.1023/A:1008912914816 (cited on page 64).

[106] Hugh R. Wilson and Jack D. Cowan. "Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons". In: *Biophysical Journal* 12.1 (Jan. 1, 1972), pp. 1–24. DOI: 10.1016/S0006-3495(72)86068-5 (cited on pages 64, 78).

[107] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. "Automatic Posterior Transformation for Likelihood-Free Inference". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, June 9–15, 2019, pp. 2404–2414 (cited on page 64).

[108] George Papamakarios, David C. Sterratt, and Iain Murray. "Sequential Neural Likelihood: Fast Likelihood-Free Inference with Autoregressive Flows". May 18, 2018 (cited on page 64).

[109] Jan-Matthis Lueckmann et al. "Flexible Statistical Inference for Mechanistic Models of Neural Dynamics". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 1289–1299 (cited on pages 64, 75).

[110] George Papamakarios and Iain Murray. "Fast ϵ-Free Inference of Simulation Models with Bayesian Conditional Density Estimation". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 1028–1036 (cited on pages 64, 75).

[111] A. Waibel et al. "Phoneme Recognition Using Time-Delay Neural Networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (Mar. 1989), pp. 328–339. DOI: 10.1109/29.21701 (cited on page 64).

[112] David Sussillo and Omri Barak. "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks". In: *Neural Computation* 25.3 (Dec. 28, 2012), pp. 626–649. DOI: 10.1162/NECO_a_00409 (cited on page 64).

[113] Omri Barak. "Recurrent Neural Networks as Versatile Tools of Neuroscience Research". In: *Current Opinion in Neurobiology*. Computational Neuroscience 46 (Oct. 1, 2017), pp. 1–6. DOI: 10.1016/j.conb.2017.06.003 (cited on page 64).

[114] Doron Haviv, Alexander Rivkind, and Omri Barak. "Understanding and Controlling Memory in Recurrent Neural Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, May 24, 2019, pp. 2663–2671 (cited on page 64).

[115] Arne F. Meyer et al. "Models of Neuronal Stimulus-Response Functions: Elaboration, Estimation, and Evaluation". In: *Frontiers in Systems Neuroscience* 10 (2017), p. 109. DOI: 10.3389/fnsys.2016.00109 (cited on page 67).

[116] Moritz Augustin et al. “Low-Dimensional Spike Rate Models Derived from Networks of Adaptive Integrate-and-Fire Neurons: Comparison and Implementation”. In: *PLOS Computational Biology* 13.6 (June 23, 2017), e1005545. DOI: 10.1371/journal.pcbi.1005545 (cited on pages 67, 69, 82).

[117] M. J. Betancourt and Mark Girolami. “Hamiltonian Monte Carlo for Hierarchical Models”. Dec. 3, 2013 (cited on pages 72, 81).

[118] Joshua H. Goldwyn and Eric Shea-Brown. “The What and Where of Adding Channel Noise to the Hodgkin-Huxley Equations”. In: *PLOS Computational Biology* 7.11 (Nov. 17, 2011), e1002247. DOI: 10.1371/journal.pcbi.1002247 (cited on page 75).

[119] Simon N. Wood. “Statistical Inference for Noisy Nonlinear Ecological Dynamic Systems”. In: *Nature* 466.7310 (Aug. 26, 2010), pp. 1102–1104. DOI: 10.1038/nature09319 (cited on page 75).

[120] Werner Horsthemke and René Lefever. *Noise-Induced Transitions: Theory and Applications in Physics, Chemistry, and Biology*. 2. print. Springer Series in Synergetics 15. Berlin: Springer, 2006. 318 pp. (cited on pages 75, 132).

[121] Liam Paninski, Jonathan W. Pillow, and Eero P. Simoncelli. “Maximum Likelihood Estimation of a Stochastic Integrate-and-Fire Neural Encoding Model”. In: *Neural Computation* 16.12 (2004), pp. 2533–2561. DOI: 10.1162/0899766042321797 (cited on page 75).

[122] Gonzalo Mena and Liam Paninski. “On Quadrature Methods for Refractory Point Process Likelihoods”. In: *Neural Computation* 26.12 (Dec. 2014), pp. 2790–2797. DOI: 10.1162/NECO_a_00676 (cited on page 75).

[123] Alexandro D. Ramirez and Liam Paninski. “Fast Inference in Generalized Linear Models via Expected Log-Likelihoods”. In: *Journal of Computational Neuroscience* 36.2 (2014), pp. 215–234 (cited on page 75).

[124] John P. Cunningham and Byron M. Yu. “Dimensionality Reduction for Large-Scale Neural Recordings”. In: *Nature Neuroscience* 17.11 (Nov. 2014), pp. 1500–1509. DOI: 10.1038/nn.3776 (cited on page 75).

[125] Michael E. Rule et al. “Neural Field Models for Latent State Inference: Application to Large-Scale Neuronal Recordings”. In: *PLOS Computational Biology* 15.11 (Nov. 4, 2019), e1007442. DOI: 10.1371/journal.pcbi.1007442 (cited on pages 75, 76).

[126] Mark Girolami and Ben Calderhead. “Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (Mar. 1, 2011), pp. 123–214. DOI: 10.1111/j.1467-9868.2010.00765.x (cited on page 75).

[127] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. “Sequential Monte Carlo Samplers”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006), pp. 411–436. DOI: 10.1111/j.1467-9868.2006.00553.x (cited on page 75).

[128] Alp Kucukelbir et al. “Automatic Differentiation Variational Inference”. In: *Journal of Machine Learning Research* 18.14 (2017), pp. 1–45 (cited on page 75).

[129] Anton V. Chizhov and Lyle J. Graham. “Efficient Evaluation of Neuron Populations Receiving Colored-Noise Current Based on a Refractory Density Method”. In: *Physical Review E* 77.1 (Jan. 17, 2008), p. 011910. DOI: 10.1103/PhysRevE.77.011910 (cited on page 78).

[130] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open Source Scientific Tools for Python*. 2001— (cited on pages 80, 85, 98).

[131] The Theano Development Team et al. “Theano: A Python Framework for Fast Computation of Mathematical Expressions”. May 9, 2016 (cited on pages 80, 85).

[132] Alexandre Iolov, Susanne Ditlevsen, and Andre Longtin. “Optimal Design for Estimation in Diffusion Processes from First Hitting Times”. In: *SIAM/ASA Journal on Uncertainty Quantification* 5 (Jan. 1, 2017), pp. 88–110. DOI: 10.1137/16M1060376 (cited on page 81).

[133] Matthew D. Hoffman and Andrew Gelman. “The No-U-turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1593–1623 (cited on page 81).

[134] Andrew Gelman et al. *Bayesian Data Analysis*. Boca Raton: CRC Press, 2014 (cited on pages 81, 103, 104, 119, 121, 141).

[135] Sean Talts et al. “Validating Bayesian Inference Algorithms with Simulation-Based Calibration”. Apr. 18, 2018. DOI: 10.48550/arXiv.1804.06788 (cited on pages 81, 124).

[136] Rutger van Haasteren. “Marginal Likelihood Calculation with MCMC Methods”. In: *Gravitational Wave Detection and Data Analysis for Pulsar Timing Arrays*. Ed. by Rutger van Haasteren. Springer Theses. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 99–120. DOI: 10.1007/978-3-642-39599-4_5 (cited on page 82).

[137] Tim P. Vogels, Kanaka Rajan, and L. F. Abbott. “Neural Network Dynamics”. In: *Annual Review of Neuroscience* 28.1 (2005), pp. 357–376. DOI: 10.1146/annurev.neuro.28.061604.135637 (cited on page 86).

[138] Alexandre René and André Longtin. “Selecting Fitted Models under Epistemic Uncertainty Using a Stochastic Process on Quantile Functions”. In: *Nature Communications* 16.1 (Oct. 23, 2025), p. 9393. DOI: 10.1038/s41467-025-64658-7 (cited on page 99).

[139] Stefan Chmiela et al. “Machine Learning of Accurate Energy-Conserving Molecular Force Fields”. In: *Science Advances* 3.5 (May 5, 2017), e1603015. DOI: 10.1126/sciadv.1603015 (cited on pages 99, 100).

[140] Mengyang Gu, Xinyi Fang, and Yimin Luo. “Data-Driven Model Construction for Anisotropic Dynamics of Active Matter”. In: *PRX Life* 1.1 (Aug. 15, 2023), p. 013009. DOI: 10.1103/PRXLife.1.013009 (cited on pages 99, 100).

[141] Keith Beven and Jim Freer. “Equifinality, Data Assimilation, and Uncertainty Estimation in Mechanistic Modelling of Complex Environmental Systems Using the GLUE Methodology”. In: *Journal of Hydrology* 249.1 (Aug. 1, 2001), pp. 11–29. DOI: 10.1016/S0022-1694(01)00421-8 (cited on page 99).

[142] Albert Tarantola. “Popper, Bayes and the Inverse Problem”. In: *Nature Physics* 2.8 (8 Aug. 2006), pp. 492–494. DOI: 10.1038/nphys375 (cited on pages 99, 100).

[143] Lorena A. Barba. *Terminologies for Reproducible Research*. Version Number: 1. 2018. DOI: 10.48550/ARXIV.1802.03311 (cited on page 100).

[144] Jinchi Lv and Jun S. Liu. “Model Selection Principles in Misspecified Models”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 76.1 (Jan. 2014), pp. 141–167. DOI: 10.1111/rssb.12023 (cited on page 100).

[145] Hsiang-Ling Hsu, Ching-Kang Ing, and Howell Tong. “On Model Selection from a Finite Family of Possibly Misspecified Time Series Models”. In: *The Annals of Statistics* 47.2 (Apr. 2019). DOI: 10.1214/18-AOS1706 (cited on page 100).

[146] Sébastien Van Bellegem and Rainer Dahlhaus. “Semiparametric Estimation by Model Selection for Locally Stationary Processes”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 68.5 (Nov. 2006), pp. 721–746. DOI: 10.1111/j.1467-9868.2006.00564.x (cited on page 100).

[147] Brian M. de Silva et al. “Discovery of Physics From Data: Universal Laws and Discrepancies”. In: *Frontiers in Artificial Intelligence* 3 (2020) (cited on pages 100, 124).

[148] Bin Yu. “Stability”. In: *Bernoulli* 19.4 (Sept. 1, 2013). DOI: 10.3150/13-BEJSP14 (cited on pages 100, 124).

[149] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or Epistemic? Does It Matter?” In: *Structural Safety*. Risk Acceptance and Risk Communication 31.2 (Mar. 1, 2009), pp. 105–112. DOI: 10.1016/j.strusafe.2008.06.020 (cited on pages 100, 124).

[150] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods”. In: *Machine Learning* 110.3 (Mar. 1, 2021), pp. 457–506. DOI: 10.1007/s10994-021-05946-3 (cited on pages 100, 124).

[151] Keith Beven and Andrew Binley. “The Future of Distributed Models: Model Calibration and Uncertainty Prediction”. In: *Hydrological Processes* 6.3 (1992), pp. 279–298. DOI: 10.1002/hyp.3360060305 (cited on page 100).

[152] Jery R. Stedinger et al. “Appraisal of the Generalized Likelihood Uncertainty Estimation (GLUE) Method”. In: *Water Resources Research* 44.12 (2008). DOI: 10.1029/2008WR006822 (cited on pages 100, 124).

[153] Keith Beven and Paul Smith. “Concepts of Information Content and Likelihood in Parameter Calibration for Hydrological Simulation Models”. In: *Journal of Hydrologic Engineering* 20.1 (Jan. 1, 2015), A4014010. DOI: 10.1061/(ASCE)HE.1943-5584.0000991 (cited on page 100).

[154] Marc C. Kennedy and Anthony O'Hagan. "Bayesian Calibration of Computer Models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.3 (2001), pp. 425–464. DOI: 10.1111/1467-9868.00294 (cited on pages 100, 124).

[155] Yarin Gal, Jiri Hron, and Alex Kendall. "Concrete Dropout". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017 (cited on pages 100, 124).

[156] Florian List et al. "Galactic Center Excess in a New Light: Disentangling the γ-Ray Sky with Bayesian Graph Convolutional Neural Networks". In: *Physical Review Letters* 125.24 (Dec. 10, 2020), p. 241102. DOI: 10.1103/PhysRevLett.125.241102 (cited on page 100).

[157] Leonid Kahle and Federico Zipoli. "Quality of Uncertainty Estimates from Neural Network Potential Ensembles". In: *Physical Review E* 105.1 (Jan. 21, 2022), p. 015311. DOI: 10.1103/PhysRevE.105.015311 (cited on pages 100, 124).

[158] Alexandre René. *EMD Model Comparison Library*. Zenodo. Mar. 2025. DOI: 10.5281/zenodo.15033190 (cited on pages 102, 137, 150).

[159] Pierpaolo De Blasi and Stephen G. Walker. "Bayesian asymptotics with misspecified models". In: *Statistica Sinica* (2013). DOI: 10.5705/ss.2010.239 (cited on page 103).

[160] Hirotugu Akaike and J. de Leeuw. "Information Theory and an Extension of the Maximum Likelihood Principle". In: *Breakthroughs in Statistics: Foundations and Basic Theory*. Ed. by Samuel Kotz and Norman L. Johnson. Vol. 1. Springer Series in Statistics, Perspectives in Statistics. New York, NY: Springer, 1992, pp. 610–624. DOI: 10.1007/978-1-4612-0919-5 (cited on pages 106, 122).

[161] H Akaike. "Information Theory and an Extension of the Maximum Likelihood Principle". In: *2nd International Symposium on Information Theory*. Ed. by Petrov, B.N. and Csaki, F. Budapest, Hungary: Akadémiai Kiadó, 1973, pp. 267–281 (cited on pages 106, 122).

[162] David J. Spiegelhalter et al. "Bayesian Measures of Model Complexity and Fit". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 64.4 (Oct. 1, 2002), pp. 583–639. DOI: 10.1111/1467-9868.00353 (cited on page 106).

[163] David Luengo et al. "A survey of Monte Carlo methods for parameter estimation". In: *EURASIP Journal on Advances in Signal Processing* 2020.1 (May 29, 2020), p. 25. DOI: 10.1186/s13634-020-00675-6 (cited on page 108).

[164] B. De Schuymer, H. De Meyer, and B. De Baets. "Cycle-Transitive Comparison of Independent Random Variables". In: *Journal of Multivariate Analysis* 96.2 (Oct. 1, 2005), pp. 352–373. DOI: 10.1016/j.jmva.2004.10.011 (cited on pages 108, 138).

[165] Bernard De Baets and Hans De Meyer. "Toward Graded and Nongraded Variants of Stochastic Dominance". In: *Perception-Based Data Mining and Decision Making in Economics and Finance*. Ed. by Ildar Batyrshin et al. Berlin, Heidelberg: Springer, 2007, pp. 261–274. DOI: 10.1007/978-3-540-36247-0_10 (cited on pages 108, 138).

[166] Daniel T. Gillespie. "The Mathematics of Brownian Motion and Johnson Noise". In: *American Journal of Physics* 64.3 (Mar. 1, 1996), pp. 225–240. DOI: 10.1119/1.18210 (cited on pages 112, 133, 134).

[167] Gloria Mateu-Figueras, Gianna S. Monti, and J. J. Egozcue. "Distributions on the Simplex Revisited". In: *Advances in Compositional Data Analysis: Festschrift in Honour of Vera Pawlowsky-Glahn*. Ed. by Peter Filzmoser et al. Cham: Springer International Publishing, 2021, pp. 61–82. DOI: 10.1007/978-3-030-71175-7_4 (cited on pages 112, 136).

[168] V. Pawlowsky-Glahn and J. J. Egozcue. "Geometric Approach to Statistical Analysis on the Simplex". In: *Stochastic Environmental Research and Risk Assessment* 15.5 (Oct. 1, 2001), pp. 384–398. DOI: 10.1007/s004770100077 (cited on pages 112, 136).

[169] G Mateu-Figueras and V Pawlowsky-Glahn. "The Dirichlet Distribution with Respect to the Aitchison Measure on the Simplex - a First Approach". In: *Proceeding of the 2nd International Workshop on Compositional Data Analysis*. Compositional Data Analysis Workshop CoDaWork'05. Girona, Oct. 2005 (cited on page 112).

[170] Sergey Koposov et al. *Joshspeagle/Dynesty: V2.1.4*. Zenodo. June 2024. DOI: 10.5281/zenodo.12537467 (cited on page 121).

[171] Sumio Watanabe. “A Widely Applicable Bayesian Information Criterion”. In: *Journal of Machine Learning Research* 14.27 (2013), pp. 867–897 (cited on page 121).

[172] Thomas Nowotny et al. “Models Wagging the Dog: Are Circuits Constructed with Disparate Parameters?” In: *Neural Computation* 19.8 (Aug. 1, 2007), pp. 1985–2003. DOI: 10.1162/neco.2007.19.8.1985 (cited on page 123).

[173] Kailai Xu and Eric Darve. “Physics Constrained Learning for Data-Driven Inverse Modeling from Sparse Observations”. In: *Journal of Computational Physics* 453 (Mar. 15, 2022), p. 110938. DOI: 10.1016/j.jcp.2021.110938 (cited on page 123).

[174] Tina Toni et al. “Approximate Bayesian Computation Scheme for Parameter Inference and Model Selection in Dynamical Systems”. In: *Journal of The Royal Society Interface* 6.31 (July 9, 2008), pp. 187–202. DOI: 10.1098/rsif.2008.0172 (cited on page 124).

[175] Sixing Chen, Antonietta Mira, and Jukka-Pekka Onnela. “Flexible Model Selection for Mechanistic Network Models”. In: *Journal of Complex Networks* 8.2 (Apr. 1, 2020), cnz024. DOI: 10.1093/comnet/cnz024 (cited on page 124).

[176] Keith Beven and Andrew Binley. “GLUE: 20 Years On”. In: *Hydrological Processes* 28.24 (Nov. 29, 2014), pp. 5897–5918. DOI: 10.1002/hyp.10082 (cited on page 124).

[177] Paul D. Arendt, Daniel W. Apley, and Wei Chen. “Quantification of Model Uncertainty: Calibration, Model Discrepancy, and Identifiability”. In: *Journal of Mechanical Design* 134.100908 (Sept. 28, 2012). DOI: 10.1115/1.4007390 (cited on page 124).

[178] Andrew Gelman, Xiao-Li Meng, and Hal Stern. “Posterior Predictive Assessment of Model Fitness Via Realized Discrepancies”. In: *Statistica Sinica* 6.4 (1996), pp. 733–760 (cited on page 124).

[179] Gemma E. Moran, John P. Cunningham, and David M. Blei. “The Posterior Predictive Null”. In: *Bayesian Analysis* 18.4 (Dec. 2023), pp. 1071–1097. DOI: 10.1214/22-BA1313 (cited on page 124).

[180] Richard M Golden. “Statistical Tests for Comparing Possibly Misspecified and Nonnested Models”. In: *Journal of Mathematical Psychology* 44.1 (Mar. 2000), pp. 153–170. DOI: 10.1006/jmps.1999.1281 (cited on page 124).

[181] Martin Modrák et al. “Simulation-Based Calibration Checking for Bayesian Computation: The Choice of Test Quantities Shapes Sensitivity”. In: *Bayesian Analysis* Advance publication (2023), p. 28. DOI: 10.1214/23-BA1404 (cited on page 124).

[182] Tuomas Sivula et al. *Uncertainty in Bayesian Leave-One-Out Cross-Validation Based Model Comparison*. Oct. 2023. DOI: 10.48550/arXiv.2008.10296 (cited on page 126).

[183] Alexandre René. *Efficient and Flexible Simulator of the Lobster Pyloric Circuit*. Version v0.1.2. Zenodo, July 22, 2024. DOI: 10.5281/zenodo.12797200 (cited on pages 126, 127, 137).

[184] Alexandre René. *Solid Colored Noise Using Sparse Convolutions*. Version v0.2.0. Zenodo, July 24, 2024. DOI: 10.5281/zenodo.12805191 (cited on pages 127, 137).

[185] J. P. Lewis. “Algorithms for Solid Noise Synthesis”. In: *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’89. New York, NY, USA: Association for Computing Machinery, July 1, 1989, pp. 263–270. DOI: 10.1145/74333.74360 (cited on page 127).

[186] Philipp Rudiger et al. *Holoviz/Holoviews: Version 1.18.1*. Version v1.18.1. Zenodo, Nov. 9, 2023. DOI: 10.5281/zenodo.10089811 (cited on page 129).

[187] Crispin W. Gardiner. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences ; with 29 Figures*. Springer Series in Synergetics 13. Berlin: Springer, 1983. 442 pp. (cited on page 132).

[188] H. Risken. *The Fokker-Planck Equation: Methods of Solution and Applications*. 2nd ed. Springer Series in Synergetics v. 18. Berlin ; New York: Springer-Verlag, 1989. 472 pp. (cited on page 132).

[189] Alexandre René. *Code for Article "Selecting Fitted Models under Epistemic Uncertainty Using a Stochastic Process on Quantile Functions"*. Zenodo. Mar. 2025. DOI: 10.5281/zenodo.15032858 (cited on pages 136, 137).

[190] Brian Conrey et al. “Intransitive Dice”. In: *Mathematics Magazine* 89.2 (Apr. 2016), pp. 133–143. DOI: 10.4169/math.mag.89.2.133 (cited on page 138).

[191] Marco Taboga. “Jeffreys’ Scale | Grades or Categories of Evidence for the Bayes Factor”. In: *Lectures on Probability Theory and Mathematical Statistics*. Kindle Direct Publishing, 2021, Online appendix (cited on page 141).

[192] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press, 1997 (cited on page 145).

[193] David Duvenaud. “Bullshit That I and Others Have Said about Neural ODEs”. Workshop presentation. NeurIPS – Workshop “Retrospectives: A Venue for Self-Reflection in ML Research” (Vancouver). Dec. 2019 (cited on page 150).

[194] Alexandre René. *Sinn*. mackelab (cited on page 150).

[195] Alexandre René. *Alcrene/Smttask*. Sept. 18, 2025 (cited on page 150).

[196] Andrew Davison. “Automated Capture of Experiment Context for Easier Reproducibility in Computational Research”. In: *Computing in Science & Engineering* 14.04 (July 1, 2012), pp. 48–56. DOI: 10.1109/MCSE.2012.41 (cited on page 150).

[197] Alexandre René. *Pyloric Network Simulator*. Zenodo, Aug. 6, 2025. DOI: 10.5281/zenodo.16751279 (cited on page 153).

[198] Marcel Stimberg, Romain Brette, and Dan FM Goodman. “Brian 2, an Intuitive and Efficient Neural Simulator”. In: *eLife* 8 (Aug. 20, 2019). Ed. by Frances K Skinner et al., e47314. DOI: 10.7554/eLife.47314 (cited on page 153).

[199] Srinivas Gorur-Shandilya, Alec Hoyland, and Eve Marder. “Xolotl: An Intuitive and Approachable Neuron and Network Simulator for Research and Teaching”. In: *Frontiers in Neuroinformatics* 12 (Nov. 26, 2018). DOI: 10.3389/fninf.2018.00087 (cited on page 153).

[200] Astrid A. Prinz and Cengiz Gunay. *Lobster STG Pyloric Network Model with Calcium Sensor*. This was originally hosted on https://biology.emory.edu/research/Prinz/database-sensors/, but that URL became unavailable in 2024. ModelDB. URL: https://modeldb.science/144387 (visited on 10/24/2025) (cited on pages 153, 158).

[201] *OpenSourceBrain/PyloricNetwork*. OpenSourceBrain, Aug. 22, 2025 (cited on page 153).

[202] James Bradbury et al. *JAX: Composable Transformations of Python+NumPy Programs*. Version 0.4.23. 2023 (cited on page 153).

Glossary

acetylcholine Neurotransmitter involved in cholinergic synapses.

action potential A rapid, localized change in a neuron's membrane potential, exhibited as a voltage pulse travelling along either the axon or dendrites. In most cases the temporal course of the pulse is stereotyped.

activation Neural network analogue to the membrane potential of a biological neuron: units with higher activation will more strongly activate downstream units. A common symbol is u. Some more abstract neuron models, such as those for populations, may in fact refer to the abstract membrane potential instead as "membrane activations". Note that this is different from the activation variables of an ion channel, which control the flow of ions but don't directly trigger a response from neighbours.

activation variable Conductance models treat each type i of ion channel as an ohmic resistor with a *voltage-dependence* conductance $g_i = m^p h g_i^{\max}$. Both m and h are within the interval $[0, 1]$ and are respectively referred to as *activation* and *inactivation* variables. Also used for generically to refer to a gating variable taking values between 0 and 1.

axon Arm-like extension of a neural cell which carries action potentials from the soma to the dendrites of downstream neurons.

batch A subset of training data used to compute one gradient update.

cholinergic Said of a synapse which uses acetylcholine as a neurotransmitter. In the lobster's pyloric circuit (c.f. Section 1.2), these have slower kinetics than glutamatergic synapses.

conductance model Model for the evolution of the membrane potential where the membrane's lipid bilayer is treated as two plates of a capacitor, connected in parallel by multiple voltage sources representing the various ion channels permeating the membrane.

consistent estimator An estimator is *consistent* if the true value is recovered in the limit of infinite data.

dendrite Tree-like extension of a neural cell which can initiate action potentials (based on inputs from upstream axons) and carry them to the soma.

empirical risk Sample estimate of the risk.

estimator Expression computed from samples of a random variable which estimates a statistic of the underlying distribution. For example, $(X_1 + \cdots + X_L)/L$ is an estimator for the mean of the random variable X. A statistic is itself a random variable.

excitatory Said of the input to a neuron which increases its propensity to initiate an action potential.

firing threshold When the membrane potential of a neuron exceeds this threshold, an action potential is initiated.

fit Process by which a model is optimized by adjusting its parameters.

generalized integrate-and-fire A more realistic form of the leaky integrate-and-fire, with a continuous increase in firing probability as the membrane potential crosses the firing threshold.

glutamate A neurotransmitter. In models, almost always treated as excitatory (i.e. as causing a depolarization of the membrane potential).

glutamatergic Said of a synapse which uses glutamate as its neurotransmitter.

gradient descent An minimization algorithm implemented by a sequence of parameter updates which locally minimize the loss based on its local gradient. Classic gradient descent uses the entire dataset at every step to compute the gradient.

hypothesis class The set functions from which a learning machine tries to select the one with the lowest loss on the training data. More general than a structural model, a hypothesis class need not be expressed as a single parametrized set of equations.

inhibitory Said of the input to a neuron which decreases its propensity to initiate an action potential.

layer (neural network) Combination of an affine transformation, parameterized by a weight matrix W and a bias vector and an element-wise nonlinear operation.

learn Process by which a learning machine is optimized by adjusting its parameters. Although equivalent to *fitting a model, the term *learn* is usually reserved for learning machines and neural networks.

learning machine Combination of a hypothesis class, loss and a learning rule. Given training data, this returns a fitted model.

loss Another word for *objective function*, with the explicit connotation that the objective is one we seek to minimize.

membrane potential Difference in electric potential across the membrane of a cell. Typically the intracellular side is around 70 mV lower than the extracellular environment. In population models, the variable interpreted as the "membrane potential" often summarizes many effects and does not correspond to a real potential; in these cases, it may be referred to instead as the "membrane activation".

model identifiability A model is *identifiable* if none of its parameters are redundant: each parameter affects the output in a unique way, such that given enough data and a perfect learning machine, all parameters can be recovered.

neural network A generic model specially designed to be easy to train by gradient descent. The Layers of a neural network are typically simple operations: they achieve their expressivity by having many parameters. May be referred to as an *artificial* neural network to distinguish from networks of biological neurons.

neurotransmitter Compound used to mediate spike transmission across a synapse. Different compounds may have different effects—either cause depolarization/excitation of the cell, or hyperpolarization/inhibition—and different chemical kinetics.

objective function A functional $l : X \to \mathbb{R}$ which, as part of a learning problem, we either seek to maximize or minimize.

risk A model's *risk* is the expectation of its loss over samples drawn from the true data-generating process.

singular model A model is *singular* if for some parameter values, its Fisher information matrix is singular (i.e. has vanishing eigenvalues). Intuitively, a model is singular if and only if its parameters are independent.

soma The main body of a neural cell, where action potentials are generated.

spike An action potential, with the connotation that we do not care about its temporal profile, but only the time of its peak. Typically represented as a vertical bar or a Dirac δ.

stochastic gradient descent A variant of gradient descent in which each update is computed using only a batch of samples drawn randomly from the dataset. In addition to being more computationally efficient, the introduced stochasticity helps to prevent the algorithm from stoping at suboptimal minima.

structural model In statistics, a *structural* model is defined by a set of equations, without specifying their parameters: it is the set of models spanned by those parameters. In machine leaning, the term hypothesis class is more common.

synapse The interface through which information (in the form of action potentials) flows from one neuron's axons to another's dendrites. There are two main types of synapses: *chemical* and *electric*. In *chemical synapses*, the arrival of an action potential triggers the release of neurotransmitters; if enough of these are picked up by a downstream dendrite, an action potential is initiated therein. *Electric synapses* are more direct, akin to spliced electric cables: the current in the downstream dendrite is directly proportional to the difference between presynaptic and postsynaptic membrane potential. Synapses may also be classified as either *excitatory* or *inhibitory*, according to whether they respectively increase or decrease the propensity of the downstream neuron to initiate an action potential (by increasing or decreasing its membrane potential).

synaptic Related to a synapse.

test set Subset of samples excluded from training and reserved exclusively for testing the final model. This allows to estimate to estimate generalization error.

train Process by which a neural network is optimized by adjusting its parameters. Although equivalent to *fitting* or *learning* a model, the term *train* is usually reserved for neural networks.

training set Subset of samples used to train a model.

Abbreviations

AB anterior bursting
AIC Akaike information criterion
BIC Bayesian information criterion
BPTT backpropagation through time
CDF Cumulative Densify Function
DIC Deviance information criterion
DNN deep neural network
EMD empirical modelling discrepancy
ERM empirical risk minimization
GLUE Generalized Likelihood Uncertainty Estimation
HB Hierarchical Beta process
KDE Kernel Density Estimate
l.h.s. left-hand side
LP lateral pyloric
MAP maximum a posteriori
MDL minimum description length
MLE maximum likelihood estimate
MST model selection test
p.d.f. probability density function
PD pyloric dilator
PPF Percent Point Function, aka quantile function
PY pyloric
r.h.s. right-hand side
RNN recurrent neural network
w.r.t. with respect to
WAIC Widely-applicable information criterion

Index