SYDE 556/750
Simulating Neurobiological Systems
Lecture 1: Introduction

Andreas Stöckel

Based on lecture notes by
Chris Eliasmith and Terrence C. Stewart

January 7, 2020

Accompanying Readings: Chapter 1 of Neural Engineering

Contents

1 Overview **1**

1.1 Overall Goal of This Course 1

1.2 Theoretical Neuroscience 1

1.3 A Useful Analogy 4

2 Neural Modelling **4**

2.1 Problems with current approaches 5

3 The Brain **5**

4 Brain structures **6**

5 The Neural Engineering Framework and the Semantic Pointer Architecture **6**

5.1 Neural Engineering Framework 6

5.2 Semantic Pointer Architecture 7

5.3 The NEF and SPA as a tool for hypothesis generation 7

5.4 The Nengo Simulator 7

6 Noise, Representation, Transformation, and Dynamics **8**

6.1 Representation 8

6.2 Transformation 10

6.3 Dynamics 10

1 Overview

Note: This entire lecture is meant to give you a broad, whirlwind overview of the material in the course. We're going to revisit most of the topics mentioned here in much more detail in the upcoming weeks.

1.1 Overall Goal of This Course

The overall goal of this course is to explore methods for modeling and simulating large-scale neurobiological systems. By large-scale we mean that we're focusing on the properties of nervous *systems* and less on the interaction between their individual components (i.e., individual neurons).

Put differently, our goal is to systematize the construction of brain models – or at least the constructions of models of brain subsystems.

So, beyond the *obvious* fact that building brain models is the most interesting thing in the world, why should you be excited about this course? What are potential applications of the material we're going to discuss?

1. **Figuring out how the brain works**
   - Health applications (better treatment of diseases).
   - Philosophical implications (new ways of understanding the mind and who we are).
2. **Building better AI systems**
   - Understanding how the brain works may help us to build better AI systems.
   - "Classical AI" (structured; symbolic) ⟷ "Deep Learning" (unstructured, "sub-symbolic", grounded). We're discussing solutions that are somewhere in between, i.e., systems that are structured but still able to learn with symbol grounding.
3. **Programming neuromorphic hardware**
   - (Spiking) neuromorphic hardware systems are inspired by brain like-information processing. The methods we discuss in this course facilitate programming this hardware.

1.2 Theoretical Neuroscience

- How does the mind work?
- Most complex and most interesting system humanity has ever studied Why study anything else?
- How should we go about studying it?

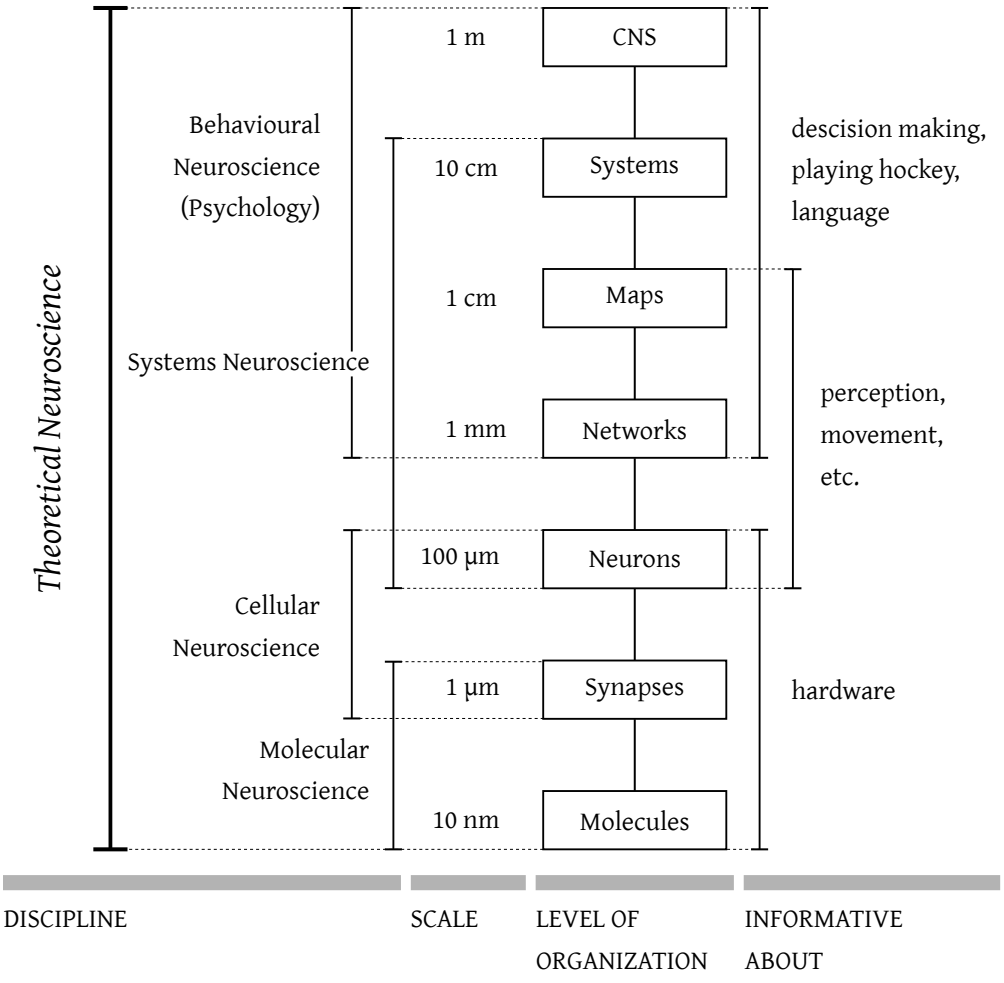


Figure 1: Levels of Organization. Adapted from ???.

- What techniques/tools?
- How do we know if we're making progress?
- How do we deal with the complexity?

1.3 A Useful Analogy

	Theoretical physics	**Theoretical neuroscience**
Quantify phenomena	$F = m\alpha$	$\hat{x} = D\alpha$
Summarize lots of data	motion of objects	neural representation of information
Speculative (generate hypotheses)	true for all velocities	true for all stimuli

Table: Comparison between theoretical physics and theoretical neuroscience. Adapted from ???

- What is Theoretical Neuroscience?
- A useful analogy is to theoretical physics
  - Similarities
    - Methods are similar
    - Goals are similar (quantification)
  - Difference
    - Central question "What exists? vs. Who are we?"
    - More simulation (because of nonlinearities) in biology

2 Neural Modelling

- Let's build it
  - Specify theory in enough detail that this is possible
  - Tends to get complex, so need computer simulation
- Bring together levels and modeling methods
  - Single neuron models (levels of detail; e.g. spikes, spatial structure, various ion channels, etc.)
  - Small network models (levels of detail; e.g. spiking neurons, rate neurons, mean fields, etc.)
  - Large network/cognitive models (levels of detail; e.g. biophysics, pure computation, anatomy, etc.)
  - Ideally allow all levels of detail below any higher level to be included as desired.
  - 'Correct' level depends on questions being asked.

2.1 Problems with current approaches

Large-scale neural models (e.g. Human Brain Project, Synapse Project, etc.)

- Lack of function or behaviour
  - Can't compare to psychological data
- Assumes canonical algorithm repeats
  - e.g., Measurements from one small part (hippocampus) are valid everywhere
  - But, different parts of the brain are very different (connectivity, cell types, inputs/outputs)
- Expects intelligence to 'emerge'
  - Unclear what 'emergence' means, how it will work, or what it explains
  - Wishful thinking?

Cognitve models (e.g. ACT-R, Soar, etc.)

- Disconnected from neuroscience, can't compare to neural data
  - Trying to map components of the model to brain areas
  - When a component is active, maybe neurons in that area are more active?
- No "bridging laws"
  - Like having rules of chemistry that never mention that it's all built out of atoms and electrons
- No constraints on the equations
  - Just anything that can be written down
  - Many possibilities; hard to figure out what matches human data best
- Maybe that's okay
  - Do we understand the brain enough to make this connection and constrain theories?
  - When understanding a word processor, do we worry about transistors?

3 The Brain

- 2 kg (2% of body weight)
- 20 Watts (25% of power consumption)
- Area: 4 sheets of paper
- Neurons: 100 billion 10^{11} (15 0000 per mm^2)
- Synapses: at least 100 trillion 10^{14} (about 1000-10000 per neuron)

4 Brain structures

5 The Neural Engineering Framework and the Semantic Pointer Architecture

Now that we've spent some time discussing other approaches aiming at large-scale modelling of neurobiological systems and pointing out their shortcomings, what is the approach at neural modelling, we're going to discuss in this course? Actually, we're going to have a look at two different modelling techniques: the *Neural Engineering Framework*, and the *Semantic Pointer Architecture*.

5.1 Neural Engineering Framework

The *Neural Engineering Framework* (NEF) has been proposed by Eliasmith and Anderson in their book "Neural Engineering" in 2003 [1]. We will spend about two thirds of this course on discussing the NEF.

The goal of the Neural Engineering framework is to treat the brain as a purely physical system and to use techniques form engineering – such as control theory, information theory, and signal processing theory – to aid a *functional* understanding of the brain in theoretical neuroscience. After all, building physical systems that perform a certain function and – to this end – *understanding* physical systems in terms of their potential function lies at the heart of engineering practice.[1] Of course, a major difference lies in the fundamental components usually employed by nature and, conversely, be engineers. Natural systems have evolved, and their individual parts, such as neurons, are thus inherently diverse and "messy". Engineers on the other hand often rely on precisely manufactured parts. So when applying engineering methods to neural modelling, we somehow have to take this messiness into account. As we will see later, we can actually use the "messiness" of neural systems to our advantage when building models.

The Neural Engineering Framework provides a systematic way to translate a vector-valued dynamical system into a spiking neural network. In other words, the NEF can be thought of as a *neural compiler*, that turns a mathematical description into a spiking neural network. Most importantly, this translation process can be informed by the vast corpus of neuroscientific data collected over the past decades. To put it differently: given a mathematical model, and a set of neurobiological constraints, including neural tuning, neuron count, connectivity, and time constants, we can create a neural network that fulfils these constraints and approximates the mathematical model.

[1] This is akin to Richard Feynman's famous chalkboard note "What I cannot create, I do not understand." [2]

5.2 Semantic Pointer Architecture

The second technique—which we're going to discuss during the last third of the course—is the *Semantic Pointer Architecture* (SPA). The SPA is a mathematical tool for building models of high level cognition. It is based on what's commonly referred to as a *Vector Symbolic Architecture* (VSA) and has been described in the book *How to Build a Brain* by Eliasmith published in 2013 [3].

VSAs provide a way to implement discrete symbolic architectures as they have been used in classical aritficial intelligence research ("Good Old Fashioned AI"; GOFAI) on top of continuous vector spaces. The SPA in particular describes how such "symbol vectors", also called "semantic pointers", can be generated from sensory input, combined—or bound to—other semantic pointers, and decompressed back into either the sensory or motor domain.

5.3 The NEF and SPA as a tool for hypothesis generation

While the NEF and SPA are independent concepts, they can be combined into a powerful tool for scientific exploration in theoretical neuroscience. In particular, the NEF provides a way to represent vector spaces on a neural substrate, and, conversely, the SPA provides a way to implement a cognitive architecture on top of vector spaces. In other words, the NEF describes the computational substrate, whereas the SPA can make use—but is not limited to—this substrate. This means that we have a powerful tool at our disposal which – looking back at fig. 1 – can be used to model biological systems from modelling synapses and neurons up to entire central nervous systems.

Of course, merely using these techniques does not magically mean that the models we build with them are good models of human cognition. In fact, depending on the kind of question we're trying to answer, our models are most definitively wrong, but that is the nature of models. The true strength of the NEF and the SPA is in aiding *hypothesis generation*.

For example, we might have an hypothesis as for how a certain cognitive task could be solved nn terms of the SPA. We furthermore know from empirical research which brain regions in are involved in solving this task, and as such have data from neuroscience. We can now use this data as constraints when mapping the model onto the brain using the NEF. The resulting spiking neural network model can then be simulated and probed in various ways. This allows us *(a)* to *verify* our hypothesis by comparing high-level behaviour data (e.g., timings and error rates) between human/animal subjects and our model and *(b)* to make *predictions* about what we should be able to measure in the case of experiments that have not been performed on real subjects.

5.4 The Nengo Simulator

In the end, the NEF and the SPA are just mathematical concepts. In order to aid scientists in pursuing the kind of hypothesis-driven research outlined above, we need tools that facili-

tate the construction of such models. The "nengo" (originally short for "*N*eural *Eng*ineering *O*bjects") neural network simulation package is such a tool. It not only provides convenient interfaces for building NEF and SPA models (as well as other kinds of artificial neural networks), but can also execute the resulting model on a variety of hard- and software-backends such GPUs and analogue and digital neuromorphic hardware.

nengo was originally developed by members of the Computational Neuroscience Research Group (CNRG) at the University of Waterloo [4]. It is now developed by a company called ABR (Applied Brain Research). nengo is available free-of-charge for non-commercial use. It can be found at `https://nengo.ai` and `https://github.com/nengo`.

Note: As part of the assignments we're going to write our own software implementation of the NEF. Correspondingly, for the sake of learning the underlying theory, you cannot use nengo at first. However, you are asked to use nengo for the last assignment, and are free to use it for the final project. So it definitively doesn't hurt to have a look at it.

6 Noise, Representation, Transformation, and Dynamics

The Neural Engineering Framework is based on three principles: Representation, Transformation, and Dynamics. We will have a closer look at each of these principles in upcoming lectures, but for now, let's quickly summarize what each of these principles is about.

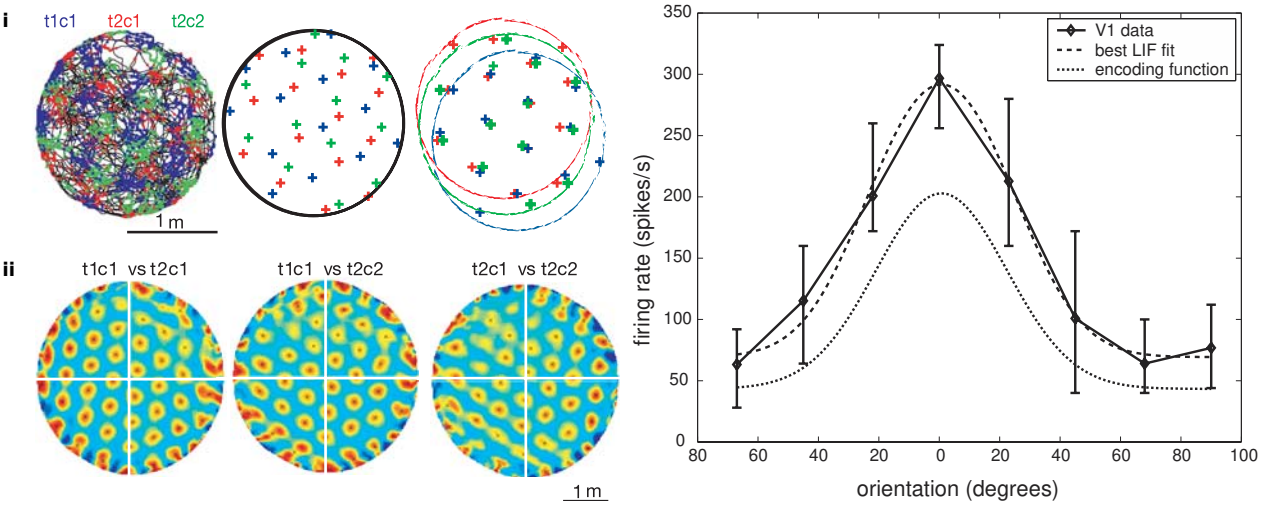
6.1 Representation

In order to solve complex tasks, nervous systems must construct *representations* of their environment. A reasonable definition of *representation* would be to say that a neuron *represents* some aspect of the environment, if this neuron is active (or conversely, exactly not active) whenever a certain stimulus is present. In other words, if there is a causal relationship between the stimulus and the neural activity. However, in general, the question of how exactly neurons represent information is still open.

In accordance with the above, we think of representations in the NEF as the mapping between a *value* that is being processed in the network and the corresponding *activity* of a *group* of neurons. Mathematically speaking, the neural activity can be seen as a vector $a(t)$ in a high-dimensional vector space, whereas the corresponding value $x(t)$ may be a situated in a lower-dimensional vector space.

For this concept of representation to be useful, we need to define a way to *decode* the represented value from the neural activity, and also a way to *encode* a value as neural activites. The NEF defines the decoding process in terms of a linear decoder, and the encoding process in terms of a nonlinear encoding.

Note that we talked representations in terms of the activity of a *population* – also referred to as an *ensemble* – of neurons, and not in terms of the activity of individual neurons. This is an



(a) Grid cells **(b)** Visual orientation tuning

Figure 2: Examples of neural representation. (a) Grid cells. Top left: trajectory of the experimental animal while moving through space. Colours (ref, blue, green) correspond to the activity of three cells recorded from the dorsocaudal medial entorhinal cortex (dMEC). Top centre: peak activity locations for each cell. Top right: the individual cell peak activities are shifted to coincide with each other, highlighting the repeating structure in each cell. Bottom: cross-correlations between the individual cell activities. Figure copied from [6], fig. 3. **(b)** Example of visual orientation tuning of a cell in primary visual cortex of a macaque monkey. Figure copied from [1], fig. 3.1.

important difference that separates the NEF from many "classical" modelling approaches in neuroscience. The NEF focuses on neuron populations as the smallest representational unit, and not individual neurons.

Example: Famous examples of neural representation are place- and grid-cells. Here neural activity is correlated with the animal being a certain location (place cells, [5]), or within a certain patch of a multi-scale hexagon grid (grid cells, [6], see fig. 2a). We're going to revisit grid cells when we talk about spatial semantic pointers. Another, more classical example is orientation tuning in primary visual cortex. Here, cells represent the orientation of line-shaped features in the visual field. See fig. 2b.

Note: When talking about representations in this course, we generally refer to "transient" activities of the network, i.e., relatively short-lived signals. Of course, longer-lived states – such as *long-term memories* – are also represented in the brain somehow, but we'll talk about this separately when discussing learning.

6.2 Transformation

Of course, merely being able to represent values within biologically plausible neural networks is not sufficient to explain complex behaviours. We need some notion of computation, i.e., the ability to compute functions $f(\mathrm{x})$ given a represented vectorial value x.

Functions are being computed by choosing the right synaptic weights that connect two populations of neurons. In a sense, we could say that the *connections between populations* compute functions.

6.3 Dynamics

References

[1] Chris Eliasmith and Charles H. Anderson. *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, Massachusetts: MIT Press, 2003. 380 pp. ISBN: 978-0-262-55060-4.

[2] Richard Feynman. *Richard Feynman's Blackboard at Time of His Death*. 1988. URL: `http://archives-dc.library.caltech.edu/islandora/object/ct1:483` (visited on 12/30/2019).

[3] Chris Eliasmith. *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford Series on Cognitive Models and Architectures. New York, New York: Oxford University Press, 2013. 456 pp. ISBN: 978-0-19-026212-9.

[4] Trevor Bekolay et al. "Nengo: A Python Tool for Building Large-Scale Functional Brain Models". In: *Frontiers in Neuroinformatics* 7.48 (2014).

[5] John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Oxford, United Kingdom: Oxford University Press, 1978. ISBN: 0-19-857206-9. URL: `http://cognitivemap.net/`.

[6] Torkel Hafting et al. "Microstructure of a Spatial Map in the Entorhinal Cortex". In: *Nature* 436.7052 (Aug. 1, 2005), pp. 801–806. ISSN: 1476-4687. DOI: `10.1038/nature03721`. URL: `https://doi.org/10.1038/nature03721`.