Computational Neuroscience

Psychology 5270 — Spring 2020

January 9, 2020

Course Information

- Class times: T 2:00-4:30, Dell 2 Rm 102
- Course websites: https://github.com/melizalab/comp-neurosci, collab
- Instructor: C Daniel Meliza (cdm8j)
- Office: Gilmer 481 (see map)
- Office Hours: by appointment
- Final: in-class presentation

Course Description

In order to survive and reproduce in a complex and highly variable physical world, animals have to extract information from sensory inputs, make decisions about how to allocate their resources, and perform tightly choreographed sequences of motor behaviors. Brains have evolved to perform these tasks reliably, flexibly, and efficiently. **Our goal is to understand how these computations are implemented by circuits of neurons**.

The challenge we face is that brains are enormously complex. Even relatively "simple" non-human animals have millions of neurons and billions of connections. Just observing a small fraction of the brain in action can generate enormous quantities of data. Voltage from an intracellular electrode is sampled 40,000 times a second, and EEGs are recorded on 19 channels sampled up to 2,000 times a second. That means almost 2.5 million data points in one minute! With so much data, how can we possibly find the information in all the noise?

The approach we'll use here is **computational modeling**, which means building toy systems that capture some aspect of how a particular system in the brain is performing its function. As we'll discuss, there are many different kinds of models and ways of building them, but one of the most useful functions of a model is that it can be used like a lens that can reveal structure in our data and help us filter out the noise.

This course will introduce you to foundational principles of computational modeling in the brain using a hands-on approach with exercises that will also build your expertise in using scientific programming to implement models and apply them to real-world data.

Course Objectives

By the end of the semester, you will be able to:

- identify different types of neuroscience data and the appropriate methods for analyzing them
- understand linear time series models and how to use them to analyze and predict neural activity
- use Python and its numerical libraries to process, analyze, and visualize neuroscience data
- think through computational problems logically and outline the steps of an analysis
- work as part of a team to write and talk about computational research
- feel confident in your ability to use computers in your research and find resources to answer

future questions

Course Schedule

This schedule is subject to change based on the interests of the class and our progress through the material. If there's something you want to cover that isn't listed, let me know! Readings and assignments will be posted to collab. The topics we will cover are as follows:

Week	Topic	Readings/Exercises
1	Introduction to computational neural models	Ewert (1974), McNaughton & Morris (1987)
2	Working with time series data	Learn some Python (see assignments)
3	Probability and spike train statistics	
4	I/O and visualization for neuroscience	
5	Open science: code and data management	Sandve et al (2013), Wilson et al (2014)
6	Linear time-invariant systems	
7	Estimating parameters and comparing models	
8	Decoding models	
9	Information theory	
10	Dynamical systems theory	Izhikevich (Ch 3–4)
11	Biophysical neuron models	Sterratt (Ch 2–3)
12	Phenomenological dynamical models	Izhikevich (Ch 5–6), Brette (2015)
13	Large-scale simulations	Stimberg et al (2019)
14–15	Finalize projects and present results	

Materials

Computing requirements

Scientific programming is essential to computational neuroscience. Digital computers are extremely good at doing the same thing over and over again, which is exactly what we need to reliably deal with the big data generated in neuroscience. In fact, everyone who works with data can benefit from being able to hand tedious tasks off to a computer. Scientific programming is an important skill for researchers at all levels, and the practice of programming will help you learn how to break down complex problems of all kinds into a set of logical steps.

Anyone who is interested in developing these skills is welcome to take the class regardless of programming experience. (Seriously!) You do need to be committed to learning these essential skills, however, which may require significant out-of-class work to nail down core concepts if you are totally new to this. I can direct you to useful tutorials and other materials as needed.

Laptops are necessary for this class. You will need to bring your laptop, running Linux, OS X, or Windows, to every class meeting. Almost all the work will be performed in Jupyter notebooks running the Python interpreter. Some exercises can be run using online resources like binder, but I strongly encourage you to install a Python environment on your laptop that you can use locally. We will spend time on the first day of class making sure everyone has a functioning environment.

Readings

There is no textbook for the course, but reviews and readings from the primary literature will be assigned throughout the semester. PDF copies will be shared through the Resources tab on the class Collab page.

Evaluation

Engagement (40%)

Programming will be taught primarily in class, so your attendance and participation during class time will directly influence what you get out of it. Throughout the semester, you will be working in class with a partner, so your engagement will also affect your partner's ability to learn and benefit from the course.

How you can succeed: Come to class with a laptop. Read or watch out-of-class material and come prepared with questions and thoughts so you can participate in discussions. Talk to your programming partner to make sure you both understand the tasks. Ask questions in class or on Piazza if anything is unclear or if you are getting an error you and your partner can't figure out.

Homework (30%)

Throughout the course, you will periodically be given take-home assignments that challenge you to think about how to break a problem into small, logical steps using pseudocode. When approaching new methods, pseudocode is a powerful way of planning your analyses without worrying about the specifics of syntax. You may write accompanying code if you wish, but that is not required or expected. Programming isn't just about making sure you've closed all your brackets; it's a new way of thinking, and these problems will give you practice with this type of thinking without your console giving you errors.

How you can succeed: Think hard about the problem and how to construct a series of explicit, logical steps that solve it. Write down your best thoughts on the solution, and read through it carefully to catch any leaps of reasoning you've made. Your process is more important than whether every step is correct. Remember, there are usually multiple ways to solve any problem.

Data Exploration (30%)

As part of a semester-long project, you and your programming partner will have the opportunity to choose one of the neuroscience data sets we'll be working with in class. As we discuss various topics in data science, you will be applying these to your data set and building toward a complete research project. Along the way, you and your partner will gain expertise on your data set and be a resource for other students in our class interested in working with that type of data. There will be milestones and chances for revision over the course of the semester. Important dates and more detailed information are laid out in the project description.

How you can succeed: Engage with the data set you have chosen. Explore the structure of your data set and read the methods section of the associated publication. Think about which methods we discuss can be applied to your data and what kinds of questions could be asked with them. Test and document functions you write to make them as easy as possible for you and the rest of the class to use. Take advantage of peer and instructor feedback to improve your final product.

Timeline

- Week 5: Choose a data set that interests you. Describe the experimental conditions and how the data were collected.
- Week 7: Write a module with functions to read raw data from the files in the data set and parse into a flexible data structure.
- Week 10: Develop a question you can answer about your data set (or a comparison between your data set and another) using one or more of the methods we have been working with.
- Week 13: Generate a set of preliminary figures that capture the most important features of your results.
- Week 14: Finalize your analyses and figures. Prepare a short presentation on your results.

Course policies

Work must be turned in on time to receive full credit. There is a penalty of 5% per day for late work unless an extension is arranged **before** the due date or an emergency prevents submission.

Due to the fact that we meet only once a week and complete much of the work in class, attendence is critical. You may miss one class without penalty. Additional absences may be excused due to religious holidays, UVA-required extracurricular activities (e.g., competitions or performances), or legitimate academic reasons; requests must be made and approved at least one week in advance.

My goal is for everyone in the class to have an equal opportunity to learn and to demonstrate their knowledge. Students with disabilities are entitled to reasonable accommodations. Contact the Student Disability Access Center (434-243-5180) for more information or to arrange accommodations.