The University of Western Ontario
Graduate Program in Neuroscience

Neuroscience 9520
Computational Modeling In Neuroscience

Fall Semester 2012

September 5, 2012

Instructor
Paul Gribble
Office / Lab: NSC 228 / 245G
paul@gribblelab.org
www.gribblelab.org
(519) 661-2111 x82237
office hours by appointment

Class Schedule
Mondays 2:00pm-3:30pm
Thursdays 11:30am-1:00pm
NSC 245A

Course Description

The goal of this one-semester graduate course is to provide students with an overview of computational models of neural systems from neuron to network to behaviour, hands-on experience using computer software to implement and test models, and the ability to critically assess original research articles in which computational modeling techniques are used to address current issues in Neuroscience research. We will cover topics such as models of single neurons (e.g. the Hodgkin-Huxley model), neural network models, supervised and unsupervised learning, and models of dynamical systems (e.g. musculoskeletal models).

Each week our meeting will be conducted as a lecture-based workshop. I will present the key material from the readings, answer student questions, and we will work through examples collaboratively. You should bring your laptop to class so you can work through examples as we go. The bulk of learning in the course will take place outside of the classroom by reading, working on weekly assignments, and by playing with toy computational models. We will be using the Python programming language (`http://www.python.org`).

Requirements and Evaluation

The requirement for this course is simple: work diligently. This includes attending class, doing the readings carefully before the seminar meets, reading beyond the syllabus, working regularly on the assignments, playing with the example code, and coming to see me for extra help, as needed. Because students have different backgrounds and interests, the amount of time necessary to master the material will vary. That being said, I am confident that every student can do so with concerted effort. One important thing to realize: We will not have enough time in the course to go over the details of every concept covered in the course, in class time. In class I will highlight the major ideas and provide a conceptual roadmap for you to navigate through the material. You will be responsible for (and you should, given your position as a graduate student, be interested in) reading the material on your own and asking questions if you need more guidance.

Each week we will meet as a class, I will present a new topic, and we will have a chance to play with some code. You are expected to do the readings in advance of each class, not afterwards. This ensures we have something to discuss as a class. Each week I will hand out an assignment, which will be due next time we meet. Feel free to work together on the assignments. Just do not hand in a document to me that is identical to someone else's. You won't learn nearly as much if you simply copy someone else's work. You will get the most out of this course if you work through the assignments on your own. Your goal here should not be to get a high grade. Your goal should be to learn as much as you can.

The final assignment is a **written report** on an original research article of your choosing, in which the authors use one or more techniques from computational neuroscience. The article you choose must be a report of primary research (not a review article) published in a mainstream peer-reviewed scientific journal. Your report should be brief (no more than 5 pages or so) and should follow the guidelines for the "Journal Club" Features in the Journal of Neuroscience[1]. In our last meeting of the term each student will also give a short **oral presentation** on their chosen article. You are responsible for reading all of the articles that will be presented that day, in advance, so that we can all discuss them together.

Finally, I am always available to answer questions you may have about the readings, the concepts, and the assignments. A non-trivial aspect of the work involved in this course will be learning how to use the computer software you choose to implement the assignments and to play with the toy examples we will be using each week to illustrate concepts.

[1] `http://www.jneurosci.org/site/misc/ifa_features.xhtml`

Class Schedule

Please refer to the course webpage for the most current schedule:

`http://www.gribblelab.org/compneuro`

Grades

There will be no exams in this course. The breakdown of grades will be as follows:

Component	Grade
Assignments	50%
Written Report	20%
Oral Presentation	20%
Participation	10%
Total	100%

Readings

Readings will be taken from a number of relevant books including: Haykin (2009); Duda et al. (2001); Mitchell (1997); McLeod et al. (1998); Shadmehr and Wise (2005); O'Reilly and Munakata (2000); Trappenberg (2010). We will also read a selection of original research articles. I encourage you to read past the assigned material to explore the topics that interest you most. I would be happy to provide you with suggestions about additional readings.

Software

The computational models we will be dealing with in this course can be readily implemented on a computer using just about any modern programming language such as C, MATLAB, Python, R, etc. There are many choices. In past years I have used MATLAB (`http://www.mathworks.com`) to teach this course. We will not be using MATLAB this time. Instead we will use Python and its associated libraries such as SciPy, NumPy, Matplotlib and Pylab.

We will have a class in which we will get Python installed on everyone's laptop, and go over a brief tutorial about the programming language itself. Throughout the course we will be implementing models in Python, and while students are not expected to have a strong programming background coming into the course, students are expected to learn some programming along the way.

A Brief Diatribe about Software

There are many reasons not to use MATLAB, and many reasons in particular not to use it in a course — reasons mainly having to do with freedom rather than technical reasons (technically speaking MATLAB is a very powerful tool). MATLAB is proprietary software, and is not free (as in beer). MATLAB is very expensive. What's more, however, neither is it free (as in speech), because you are restricted in how you use it even after you have purchased it [2].

Mathworks demands not only that you give them thousands of dollars to purchase a copy [3], but that each time they release new versions (including bug fixes), you must send them more money to maintain a license to use the software. You are restricted to only use one copy of MATLAB per license. The software is linked to the machine it's installed on and so cannot (easily) be moved to a new machine. You may not run (or even install) MATLAB on more than one machine (e.g. a desktop and a laptop) without purchasing additional licenses. Information about your computer is sent to Mathworks when you license your copy of MATLAB. If you copy MATLAB to a new machine, or give your friend a copy of MATLAB, you become a criminal because you have violated the terms of their license.

These days there is little reason to put up with this nonsense. There are many free (both as in beer, and as in speech) software environments that offer much of the functionality of MATLAB. Some examples (with blurbs from their respective websites) include:

- **GNU Octave** (`http://www.gnu.org/software/octave`) is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is *mostly* compatible with MATLAB.
- **R** (`http://www.r-project.org`) is a free software environment for statistical computing and graphics. It is rapidly becoming very popular for data analysis and graphical visualization. We use **R** in my Statistics for Neuroscience (Neuroscience 9506) course.
- **Python** (`http://www.python.org`) is a high-level programming language that is very popular. Google uses Python for many tasks including the backends of web apps such as Google Groups, Gmail, and Google Maps, as well as for some of its search-engine internals [4].

[2] The Free Software Foundation (`http://www.fsf.org/about`) provides more information about these issues not just for MATLAB but for proprietary software licenses in general.

[3] There is a Student Version of MATLAB at reduced cost, but it is hamstrung by many limitations compared to the "Professional Version" `http://www.mathworks.com/academia/student_version/`

[4] `http://en.wikipedia.org/wiki/List_of_Python_software#Commercial_uses`

SciPy (`http://www.scipy.org`) (pronounced "Sigh Pie") is an open-source **Python** library for mathematics, science, and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with **NumPy** arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers. The **matplotlib** library (`http://matplotlib.sourceforge.net/`) tries to duplicate MATLAB-style interactive plotting and seems to do a reasonable job.

- **Sage** (`http://www.sagemath.org/`) is a free, open-source mathematics software system that is meant to be a viable alternative to high-end symbolic and numerical mathematics environments like Mathematica and Maple. It is based on **Python**. It uses the "notebook" paradigm for constructing mathematical documents that Mathematica is based on. This seems mainly useful for symbolic math applications, perhaps less useful for data-intensive applications.

- The **C** language is a decades-old tried and true programming language that still remains a universal standard in computing. The GNU C Compiler (GCC, `http://gcc.gnu.org/`) is free and open-source, as is the **GNU Scientific Library** (GSL, `http://www.gnu.org/software/gsl/`), a library of over 1000 functions providing a wide range of mathematical routines such as random number generators, optimization, statistics, differential equations, least-squares fitting, and many more. For the Mac, Apple provides their in-house development environment, called **Xcode** [5], free to all users. It supports C (and other languages) and includes a ton of documentation. I run a **C Programming Boot Camp** (`http://www.gribblelab.org/CBootCamp`) in the summers, the notes are all online.

There are a number of other possibilities, but the languages listed above are good choices in that they are stable, platform-independent, used by many, and free (as in beer and as in speech).

This Course

As we go forward in the course, we will be using Python (and potentially other programming languages) to demonstrate principles and models. Note however that you are free to use whatever package you wish in assignments and throughout the course, including MATLAB if you really wish.

[5] `http://developer.apple.com/technologies/tools/xcode.html`

Choose a Programming Language

We will be using Python in the course. If however you wish to use something else, here is a guide to how to choose.

If you or your supervisor already has a MATLAB license, and the idea of supporting (by using) proprietary software doesn't bother you, then MATLAB is a reasonable choice. If you want a free alternative that is most like MATLAB, try Octave. If you want to invest some more time and effort in learning a language that will give you a wider set of skills (and be applicable to a wider set of problems, not just computational neuroscience), then choose Python/SciPy. If you want to learn the best language for speed, portability and you are a hacker at heart, choose C & the GSL.

Note: None of these languages can be "picked up" in an afternoon (unless you are already a proficient programmer, in which case all languages are basically similar). It takes some time and effort to learn a programming language, and it takes time and effort to learn to think algorithmically. Those of you with any background in computer science (whether it's formal coursework or personal hacking), will have a head start. Those of you who don't, please come to see me, I can suggest some good introductory material that will help get you started thinking like a computer programmer.

Although you don't need to become an expert programmer to benefit from this course, the more you are willing to pick up programming as a skill, the better off you will be in the long run.

Writing Reports

For generating reports, students usually attempt to use Microsoft Turd, usually on a Microsoft Windows system. This typically turns out to be problematic and frustrating for several reasons. A great alternative that is widely used is the LaTeX(pronounced "lay-tek") document typesetting system (`http://www.latex-project.org/`).

There are many reasons to use LaTeX, but some include that it is 100% platform-independent, it is free (as in beer), it is free (as in speech), and for decades it has been the absolute gold standard in document preparation software in technical fields such as computer science, physics, and mathematics. I wrote my Ph.D. thesis using LaTeX. This syllabus was prepared using LaTeX.

There is a list of LaTeX-related links on my webpage [6]. If you would like some help getting familiar with LaTeX, do let me know, I'd be happy to help. We can include a short LaTeX-related intro and tutorial sometime at the beginning of the semester if there is interest.

[6] `http://www.gribblelab.org/links`

Unsolicited Advice about Operating Systems

Finally, in general I recommend against using Microsoft Windows as your operating system. There are many reasons for this including primarily security and stability, although I gather Windows 7 represents somewhat of an improvement over previous incarnations of Windows.

A simple piece of information: Google has recently banned ALL Google employees (numbering in the tens of thousands) from using Microsoft Windows on *any* company computers (or indeed any computers at all connected to the company's network). At Google, those employees who wish to continue using Windows on their machine need clearance from "quite senior levels" [7]. Instead they recommend Linux [8] or Mac OS X [9], which are both unix-based operating systems.

If you would like to talk to me about the pros and cons of various operating systems for science, I'd be happy to discuss it.

[7] `http://www.google.com/search?&q=google+bans+microsoft+windows`

[8] `http://en.wikipedia.org/wiki/Linux`. I recommend Ubuntu `http://www.ubuntu.com/desktop` or Debian `http://www.debian.org/`

[9] `http://www.apple.com/macosx`

References

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern classification*. Wiley, New York, 2nd edition.

Ekeberg, O., Wallén, P., Lansner, A., Tråvén, H., Brodin, L., and Grillner, S. (1991). A computer based model for realistic simulations of neural networks. i. the single neuron and synaptic interaction. *Biol Cybern*, 65(2):81–90.

Haykin, S. S. (2009). *Neural networks and learning machines*. Prentice Hall, New York, 3rd edition.

Hodgkin, A. L. and Huxley, A. F. (1990). A quantitative description of membrane current and its application to conduction and excitation in nerve. 1952. *Bull Math Biol*, 52(1-2):25–71; discussion 5–23.

McLeod, P., Plunkett, K., and Rolls, E. T. (1998). *Introduction to connectionist modelling of cognitive processes*. Oxford University Press, Oxford.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

O'Reilly, R. C. and Munakata, Y. (2000). *Computational explorations in cognitive neuroscience: understanding the mind by simulating the brain*. MIT Press, Cambridge, Mass.

Shadmehr, R. and Wise, S. P. (2005). *The computational neurobiology of reaching and pointing: a foundation for motor learning*. MIT Press, Cambridge, Mass.

Trappenberg, T. P. (2010). *Fundamentals of computational neuroscience*. Oxford University Press, Oxford, 2nd edition.

van Leeuwen, J. L. (1999). Neuromuscular control: introduction and overview. *Philos Trans R Soc Lond B Biol Sci*, 354(1385):841–7.