CNS*2020 Online

TYPE - TUTORIAL • JULY 18 • SATURDAY

T Tutorial

JULY 18 • SATURDAY

12:00pm – 3:00pm

T **Characterizing neural dynamics using highly comparative time-series analysis**

Speakers: Ben D. Fulcher

Ben D. Fulcher

T7: Massive open datasets of neural dynamics, from microscale neuronal circuits to macroscale population-level recordings, are becoming increasingly available to the computational neuroscience community. There are myriad ways to quantify different types of structure in the univariate dynamics of any individual component of a neural system, including methods from statistical time-series modeling, the physical nonlinear time-series analysis literature, and methods derived from information theory. Across this interdisciplinary literature of thousands of time-series analysis methods, each method gives unique information about the measured dynamics. However, the choice of analysis methods in any given study is typically subjective, leaving open the possibility that alternative methods might yield better understanding or performance for a given task.

In this tutorial, I will introduce highly comparative time-series analysis, implemented as the software package *hctsa*, which partially automates the selection of useful time-series analysis methods from an interdisciplinary library of over 7000 time-series features. I will demonstrate how *hctsa* can be used to extract useful information from various neural time-series datasets. We will work through a range of applications using fMRI (mouse and human) and EEG (human) time-series datasets, including how to: (i) determine the relationship between structural connectivity and fMRI dynamics in mouse and human; (ii) understand the effects of targeted brain stimulation using DREADDs using mouse fMRI; and (iii) classify seizure dynamics and extract sleep-stage information from EEG.

Tutorial Website

Software tools

[1] If you want to play along at home, you can read the README and install the *hctsa* software package (Matlab): https://github.com/benfulcher/hctsa

[2] *hctsa* documentation: https://hctsa-users.gitbook.io/hctsa-manual/

References and background reading

[1] B.D. Fulcher, N. S. Jones. *hctsa*: A computational framework for automated time-series phenotyping using massive feature extraction. Cell Systems 5(5): 527 (2017). https://doi.org/10.1016/ j.cels.2017.10.001

[2] B.D. Fulcher, M.A. Little, N.S. Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *J. Roy. Soc. Interface* 10, 20130048 (2013). https://doi.org/10.1098/rsif.2013.0048

4:00pm – 7:00pm T **The use of Keras with Tensor Flow applied to neural models and data analysis**

Speakers: Cecilia Jarne

Cecilia Jarne

T5: This tutorial will help participants implement and explore simple neural models using Keras [1] as well as the implementation of neural networks to apply Deep learning tools for data analysis. It will include an introduction to modeling and hands-on exercises. The tutorial will focus on using Keras which is an open-source framework to develop Neural Networks for rapid prototyping and simulation with TensorFlow [2] as backend. The tutorial will show how models can be built and explored using python. The hands-on exercises will demonstrate how Keras can be used to rapidly explore the dynamics of the network.

Keras is a framework that greatly simplifies the design and implementations of Neural Networks of many kinds (Regular classifiers, Convolutional Neural Networks, LSTM among others). In this mini-course we will study implementations of neural networks with Keras split into two sections: On one side we will introduce the main features of Keras, showcasing some examples; and in then we will do a set of two guided on-line hands-on with exercises to strengthen the knowledge.

Tutorial Website
For this tutorial, you will need basic knowledge of NumPy, SciPy, and matplotlib. To be able to carry out the tutorial, students need a laptop with Linux and these libraries installed:

- Python
- Numpy
- SciPy
- Matplotlib
- Scikit learn
- TensorFlow
- Keras

I recommend the following sites where is explained the installation of following packages that include a set of the named libraries and some additional tools:

- https://www.anaconda.com/distribution/
- https://www.tensorflow.org/install/
- https://keras.io/

[1] Francois Chollet et al. Keras. https://keras.io, 2015.
[2] Martín Abadi, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

4:00pm – 7:00pm

T **Tools and techniques to bridge the gap between models and closed-loop neuroscience experiments**
Speakers: Pablo Varona, Rodrigo Amaducci, Manuel Reyes-Sanchez
Pablo Varona, Manuel Reyes Sanchez , Rodrigo Amaducci

T3: Models in computational neuroscience are typically used to reproduce and explain experimental findings, to draw new hypotheses from their predictive power, to undertake the low observability of the brain, etc. However, computational models can also be employed to interact directly with living nervous systems, which is a powerful way of unveiling key neural dynamics by combining experimental and theoretical efforts. However, protocols that simultaneously combine recordings from living neurons and input/outputs from computational models are not easy to design or implement. In this tutorial, we will describe several tools and techniques to build such kind of open and closed-loop interactions: from basic dynamic-clamp approaches to build hybrid circuits to more complex configurations that can include several interacting living and artificial elements. We will emphasize the need of open-source real-time software technology for some of these interactions.

In particular, we will focus on two software packages that can implement closed-loop interactions between living neurons and computational neuroscience models. The first one, RTHybrid, is a solution to build hybrid circuits between living neurons and models. This program, developed by the organizers, includes a library of neuron and synapse models and different algorithms for the automatic calibration and adaptation of hybrid configurations. The second software tool, RTXI, allows to program specific modules to implement a wide variety of closed-loop configurations and includes many handy modularization and visualization tools. Both programs can be used in very wide contexts of hybrid experimental design and deal with real-time constraints. During the tutorial, we will show how to install and use these programs in standard computer platforms, and we will provide attendees the possibility of building and testing their first designs.

Tutorial Website

Software tools

- RTHybrid: https://github.com/GNB-UAM/RTHybrid
- RTXI: http://rtxi.org/

4:00pm – 10:00pm T **Building mechanistic multiscale models, from molecules to networks, using NEURON and NetPyNE**

Speakers: Salvador Dura-Bernal, Robert A. McDougal, William W. Lytton

Salvador Dura-Bernal, Robert A McDougal, William W Lytton

T2: Understanding brain function requires characterizing the interactions occurring across many temporal and spatial scales. Mechanistic multiscale modeling aims to organize and explore these interactions. In this way, multiscale models provide insights into how changes at molecular and cellular levels, caused by development, learning, brain disease, drugs, or other factors, affect the dynamics of local networks and of brain areas. Large neuroscience data-gathering projects throughout the world (e.g. US BRAIN, EU HBP, Allen Institute) are making use of multiscale modeling, including the NEURON ecosystem, to better understand the vast amounts of information being gathered using many different techniques at different scales.

This tutorial will introduce multiscale modeling using two NIH-funded tools: the NEURON simulator [1], including the Reaction-Diffusion (RxD) module [2,3], and the NetPyNE tool [4]. The tutorial will include background, examples, and hands-on exercises covering the implementation of models at four key scales: (1) intracellular dynamics (e.g. calcium buffering, protein interactions), (2) single neuron electrophysiology (e.g. action potential propagation), (3) neurons in extracellular space (e.g. spreading depression), and (4) networks of neurons. For network simulations, we will use NetPyNE, a high-level interface to NEURON supporting both programmatic and GUI specification that facilitates the development, parallel simulation, and analysis of biophysically detailed neuronal circuits. We conclude with an example combining all three tools that links intracellular molecular dynamics with network spiking activity and local field potentials. Basic familiarity with Python is recommended. No prior knowledge of NEURON or NetPyNE is required, however, participants are encouraged to download and install each of these packages prior to the tutorial.

TUTORIAL WEBSITE

Tools:

- NEURON: neuron.yale.edu/
- RxD: neuron.yale.edu/neuron/docs/reaction-diffusion
- NetPyNE: netpyne.org

References:

1. Lytton WW, Seidenstein AH, Dura-Bernal S, McDougal RA, Schürmann F, Hines ML. Simulation Neurotechnologies for Advancing Brain Research: Parallelizing Large Networks in NEURON. Neural Comput. 28, 2063–2090, 2016.
2. McDougal R, Hines M, Lytton W. (2013) Reaction-diffusion in the NEURON simulator. Front. Neuroinform. 7, 28. 10.3389/fninf.2013.00028
3. Newton AJH, McDougal RA, Hines ML and Lytton WW (2018) Using NEURON for Reaction-Diffusion Modeling of Extracellular Dynamics. Front. Neuroinform. 12, 41. 10.3389/fninf.2018.00041
4. Dura-Bernal S, Suter B, Gleeson P, Cantarelli M, Quintana A, Rodriguez F, Kedziora DJ, Chadderdon GL, Kerr CC, Neymotin SA, McDougal R, Hines M, Shepherd GMG, Lytton WW. (2019) NetPyNE: a tool for data-driven multiscale modeling of brain circuits. eLife 2019;8:e44494

4:00pm – 10:00pm T **New interfaces for teaching with NEST: hands-on with the NEST Desktop GUI and NESTML code generation**

Speakers: Sebastian Spreizer, Charl Linssen, Renato Duarte

Charl Linssen, Sebastian Spreizer, Renato Duarte

T1: NEST is established community software for the simulation of spiking neuronal network models capturing the full detail of biological network structures [1]. The simulator runs efficiently on a range of architectures from laptops to supercomputers [2]. Many peer-reviewed neuroscientific studies have used NEST as a simulation tool over the past 20 years. More recently, it has become a reference code for research on neuromorphic hardware systems [3].
This tutorial provides hands-on experience with recent improvements of NEST. In the past, starting out with NEST could be challenging for computational neuroscientists, as models and simulations had to be programmed using SLI, C++ or Python. NEST Desktop changes this: It is an entirely graphical approach to the construction and simulation of neuronal network models. It runs installation-free in the browser and has proven its value in several university courses. This opens the domain of NEST to the teaching of neuroscience for students with little programming experience.
NESTML complements this new interface by enhancing the development process of neuron and synapse models. Advanced researchers often want to study specific features not provided by models already available in NEST. Instead of having to turn to C++, using NESTML they can write down differential equations and necessary state transitions in the mathematical notation they are used to. These descriptions are then automatically processed to generate machine-optimised code.
After a quick overview of the current status of NEST and upcoming new functionality, the tutorial works through a concrete example [4] to show how the combination of NEST Desktop and NESTML can be used in the modern workflow of a computational neuroscientist.

References

1. Gewaltig M-O & Diesmann M (2007) NEST (Neural Simulation Tool) Scholarpedia 2(4):1430
2. Jordan J., Ippen T., Helias M., Kitayama I., Sato M., Igarashi J., Diesmann M., Kunkel S. (2018) Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers. Frontiers in Neuroinformatics 12: 2
3. Gutzen R., von Papen, M., Trensch G., Quaglio P. Grün S., Denker M. (2018) Reproducible Neural Network Simulations: Statistical Methods for Model Validation on the Level of Network Activity Data. Frontiers in Neuroinformatics 12 (90)
4. Duarte R. & Morrison A. (2014). “Dynamic stability of sequential stimulus representations in adapting neuronal networks”, Front. Comput. Neurosci.

7:00pm – 10:00pm T **Methods from Data Science for Model Simulation, Analysis, and Visualization**

Speakers: Cengiz Gunay, Anca Doloc-Mihu

Cengiz Gunay, Anca Doloc-Mihu

T6: Computational neuroscience projects often involve a large number of simulations for parameter search of computer models, which generates a large amount of data. With the advances in computer hardware, software methods, and cloud computing opportunities making this task easier, the amount of collected data has exploded, similar to what has been happening in many fields. High-performance computing (HPC) methods have been used in the computational neuroscience field for a while. However, the use of novel data science and big data methods are less frequent. In this tutorial, we will review established HPC methods and introduce novel data science tools to be used in computational neuroscience workflows, starting from the industry standard of Apache Hadoop (https://hadoop.apache.org/) to newer tools, such as Apache Spark (https://spark.apache.org/). These tools can be used for either model simulation or post-processing and analysis of the generated data. To visualize the data, we will review novel web-based interactive dashboard technologies mostly based on Javascript and Python.

7:00pm – 10:00pm T **Neuromorphic VLSI realization of the Hippocampal formation**

Speakers: Anu Aggarwal

Anu Aggarwal

T4: Neuromorphic circuits are inspired by the organizing principles of biological neural circuits. These designs implement the computational neuroscience models of different parts of the brain in silicon. These silicon devices can perform actual work unlike the computer models. One of the main reasons for interest in this field is that the electrical and computer engineers wish to implement the superior processing powers of the brain to build machines like computers. For similar processing power, brain consumes much less power than a computer. Thus, scientists are interested in building power-efficient machines that are based on brain algorithms. Neuromorphic architectures often rely on collective computation in parallel networks. Adaptation, learning and memory are implemented locally within the individual computational elements as opposed to separation between memory and computations in conventional computers. As the Moore's law has hit the limits, there is interest in brain-inspired computing to build small, and power efficient computing machines. Application domains of neuromorphic circuits include silicon retinas, cochleas for machine vision and audition, real-time emulations of networks of biological neurons, the lateral superior olive and hippocampal formation for the development of autonomous robotic systems and even replacement of brain neuronal functions with silicon neurons. This tutorial covers introduction to silicon Neuromorphic design with example of silicon implementation of the hippocampal formation.

Tutorial Workshop

Presentations/Lectures

1. Brief background of Neuromorphic VLSI design, anatomy and physiology (including lab experimental data) of the Hippocampal formation
2. Computational Neuroscience Models of the Hippocampal formation
3. VLSI design or silicon realization of the Hippocampal formation

Background readings (not required)

1. Analog VLSI and Neural systems by Carver Mead, 1989
2. J. O'Keefe, 1976, "Place units in the hippocampus of the freely moving rat", Exp. Neurol. 51, 78-109.
3. J. S. Taube, R. U. Muller, J. B Ranck., Jr., 1990a, "Head direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis", J Neurosci., 10, 420-435.
4. J. S. Taube, R. U. Muller, J. B Ranck., Jr., 1990b, "Head direction cells recorded from the post-subiculum in freely moving rats. II. Effects of environmental manipulations", J Neurosci., 10, 436-447.
5. T. Hafting, M. Fyhn, S. Molden, M. B. Moser., E. I. Moser, August 2005, "Microstructure of a spatial map in the entorhinal cortex", Nature, 436, 801-806.
6. B. L. McNaughton, F. P. Battaglia, O. Jensen, E. I. Moser & M. B. Moser, 2006, "Path integration and the neural basis of the 'cognitive map'", Nature Reviews Neuroscience, 7, 663-678.
7. H. Mhatre, A. Gorchetchnikov, and S. Grossberg, 2012, "Grid Cell Hexagonal Patterns Formed by Fast Self-Organized Learning within Entorhinal Cortex", Hippocampus, 22:320–334.T. Madl, S. Franklin, K. Chen, D. Montaldi, R. Trappl, 2014, "Bayesian integration of information in hippocampal place cells", PLOS one, 9(3), e89762.
8. Aggarwal, 2015, "Neuromorphic VLSI Bayesian integration synapse", the Electronics letters, 51(3):207-209.
9. A.Aggarwal, T. K. Horiuchi, 2015, "Neuromorphic VLSI second order synapse", the Electronics letters, 51(4):319-321.
10. A.Aggarwal, 2015, "VLSI realization of neural velocity integrator and central pattern generator", the Electronics letters, 51(18), DOI: 10.1049/el.2015.0544.
11. A.Aggarwal, 2016, "Neuromorphic VLSI realization of the Hippocampal Formation", Neural Networks, May; 77:29-40. doi: 10.1016/j.neunet.2016.01.011. Epub 2016 Feb 4.