Computational Methods in Stochastics

Preliminaries

Contents

1. Simulating standard probability distributions.
2. Methods of simulating 'non-standard' distributions. Logarithmic binning.
3. Markov processes and stochastic models.
4. Monte Carlo (MC) method and Metropolis sampling.
5. Markov Chain Monte Carlo (MCMC) method; Gibbs and Metropolis-Hastings sampling.
6. Hamiltonian/Hybrid Monte Carlo (HMC) method.

The course is completed by doing programming assignments - and taking the exam. We’ll see about the exam this year…

Motivation

Theory

The purpose of this course is to familiarise you with the fundamental (mathematical) principles and notation of stochastic processes and methods used to simulate them.

The goal is for you to be able to read the literature ‘fluently’, which means that you have sufficiently good command of the notation and basic principles.

Practise

After this course you should be able to implement central stochastic simulation methods, use them, and also derive variations of them.

Exercises

The course is completed by doing the programming exercises (and possibly taking the exam). Grading is 0 - 5.

The grade was determined by the exercises (appr. 70 %) and the exam (appr. 30 %). In this Covid situation, should we just settle with the assignments? One assignment typically involves implementing an algorithm and using it to produce the requested results.

The programming language is **Python**. This is the trending language in data science and machine learning. If you haven't used it, this is a good place to learn it.

When you are asked to implement fundamental methods, you are not allowed to just use high-level library functions to do the job. That way you wouldn't learn too much…

Exercises

Since the purpose is for you to learn and understand (and possibly even enjoy yourselves), you can openly collaborate doing the exercises. Still, each of you is expected to hand in his/her own individual 'report'.

A report consists of the algorithm with sufficient comments, so that it can be checked and results, which typically include some computed values and graphs. Answers to the questions should be argued for – with equations if needed.

There are weekly deadlines for returning the reports.

Lectures

We can go through the material in a way that makes sense. “Traditional” lectures are fine by me. Alternatively, everyone might have a look at the material and assignments first and we could discuss tough parts. Or a combination of these two… Let’s see what works best.

Literature

The course is based mainly on three books:

1. **Mark A. Pinsky, Samuel Karlin: An Introduction to Stochastic Modeling (2011 Elsevier).** (The older version, which is available in the library will do just as well. Curiously, one of the original authors was removed from the new print. The older print is: Howard M. Taylor, **Samuel Karlin, An Introduction to Stochastic Modeling (1998 Academic Press).**)
2. **Darren J. Wilkinson: Stochastic Modelling for Systems Biology, 2012 CRC Press.** (We will not cover any systems biology stuff, but merely use this excellent book to learn stochastic methods. The book is available in the CS library.)
3. **Hossein Pishro-Nik:** Probability, Statistics, and Random Processes. This book is available online along with some lecture slides and videos. It is by far pedagogically the best of these three books and references to the relevant parts in the book will be made in the lecture slides. Under the current lock-down circumstances it is my sincere hope that it will also make my lectures somewhat redundant.

Literature

In addition parts of the review on MCMC methods by Radford M. Neal will be used.

You may manage ok just by studying the lecture notes, but reading the relevant parts in the books and the review is highly recommended.

Programming

The assignments are to be programmed in python using Jupyter Notebooks. Return the Notebooks in which you made the assignments in MyCourses. Write enough comments (text cells) so as to keep the assistants workload reasonable. All the documentation for python that you can possibly need in this course can be found in Jupyter Notebook's help. Very relevant stuff can also be found in the following book:
Wes McKinney: Python for Data Analysis (2018 O'Reilly).

Electronic version of this book can be read in the Aalto library learning platform:
https://aalto.finna.fi/Record/alli.861295
The news and instructions can be found in:
https://www.aalto.fi/en/news/safari-tech-books-online-is-now-oreilly-safari-learning-platform

Installation of Python Interpreter

The free Anaconda distribution:

Windows:

Download the Anaconda installer (https://www.anaconda.com/download/#macos) and follow the instructions on this download page. you can start the interpreter by typing **python** and exit by typing Ctrl-Z.

Mac:

Download the OS X Anaconda installer (the same place as above), which should be named something like *Anaconda3-4.1.0-MacOSX-x86_64.pkg*. Double-click the *.pkg* file to run the installer. When the installer runs, it automatically appends the Anaconda executable path to your *.bash_profile* file. This is located at */Users/$USER/.bash_profile*. To verify everything is working, try launching IPython in the system shell (open the Terminal application to get a command prompt): $ ipython To exit the shell, press Ctrl-D or type **exit()** and press Enter.

Installation of Python Interpreter

The free Anaconda distribution:

GNU/Linux

Python interpreter is installed in the university computers. If you have a Linux laptop, then see the detailed description in McKinney: Python for Data Analysis: 1.4. Installation and Setup.

Installing or updating Python packages can in Apple be done via terminals in a Linux way (commands like **conda install** and **conda update**). Instructions for this in all operating systems can be found on the Anaconda page.

About Python

Python is an *interpreted* language - as opposed to languages where all are commands/statements are *compiled* and then the whole algorithm is executed. The interpreter runs a program one statement at a time. For learning and trying out things an interpreted language is excellent. However, one pays for this convenience in longer execution times.

In, for example, C loops run fast (small overhead). In Python the overhead of loops is big. Consequently, for efficiency everything should be vectorised (computed in matrix form) in Python. In this course we are interested in principle and don't care about efficiency, so writing loops instead of matrices is fine.

If you want to go through some basics of python and the interpreter, see e.g. **McKinney's book**, Chapter 2.

Jupyter Notebook

Jupyter notebook is an easy way of running Python, make notes and comments etc.

It can be started by running the comman **jupyter** in a terminal.

On many platforms Jupyter will open up in your default browser.

A good option is to launch Anaconda-Navigator and start Jupyter therein. Then you can easily change the environment you run your applications in, read the documentation or developer blog, etc.

Jupyter notebook files are an easy way to do and send reports of the assignments in this course.

Essential Python Libraries

NumPy: Numerical Python

To use:
import numpy as np

("as …" is optional: for example arrays would be numpy.array(…) or np.array(…))

For example, NumPy operations perform complex computations on entire arrays without the need for Python for loops.

pandas:

To use:
import pandas as pd

pandas provides high-level data structures and functions designed to make working with structured or tabular data fast, easy, and expressive. pandas blends the high-performance, array-computing ideas of NumPy with the flexible data manipulation capabilities of spreadsheets and relational databases (such as SQL).

Essential Python Libraries

matplotlib:

To use:
import matplotlib.pyplot as plt

Library for producing plots and other two-dimensional data visualizations.

(random:

module for random number generation)

<u>Libraries that will be useful later, but not necessary in this course:</u>

SciPy

A collection of packages addressing a number of different standard problem domains in scientific computing. For example, scipy.integrate, scipy.optimize.

scikit-learn

A general-purpose machine learning toolkit for Python programmers.

statsmodels

A statistical analysis package.

Online Python Stuff

Online Courses & Documentation

Tutorials, Python courses:
https://www.python-course.eu/index.php

Python for Science:
http://kestrel.nmt.edu/~raymond/software/python_notes/index.html

matplotlib: https://matplotlib.org

Python Documentation:
https://docs.python.org/3.6/contents.html

IPython Documentation:
https://ipython.readthedocs.io/en/stable/

Online Python Stuff

SciPy Lecture Notes:
http://www.scipy-lectures.org/index.html

Python Data Science Handbook:
https://jakevdp.github.io/PythonDataScienceHandbook/

… and there's a whole lot more to find by googling.