LTCC Proposed Course

Title: Principles and Design Patterns in Data Scientific Software Engineering

Basic Details:

- Core Audience (1styr or 2nd/3rdyr: pure, app. or stats): PhD students at any level, application/stats interest
- Course Format (**Extended**: 5 x 2hr lectures or **Intensive**: 2 x 4hr lectures over 2 consecutive days): extended

Course Description:

- Keywords: data scientific software, software engineering, open source, design patterns, scikit-learn-like libraries, object oriented programming, AI frameworks

- Syllabus:

Data scientific software frameworks such as scikit-learn or mlr are workhorses of modern AI and data scientific modelling practice. Designing data scientific software frameworks with expressive declarative syntax and clear software structure is a task which requires simultaneous consideration of best software engineering practice, mathematical formalism, and methodological understanding of the data scientific area. Research into construction and design principles of data scientific software is a relatively new field, with the number of open source framework toolboxes for advanced tasks growing rapidly over the last years.

This course gives an introduction to the varied methodological background of the field, alongside exercises providing a gentle introduction into collaborative coding practices of modern open source communities, with opportunities for mentored contributions to existing open source tools, or creation of new software.

The course is equally suited for theorists or methodologists who would like to build their software engineering skills, as well as programmers seeking an introduction to formal aspects of data scientific software frameworks.

Detail syllabus:

1. Introduction to supervised learning and the mathematical/statistical setting
    - Quick intro/recap to joint and conditional random variable notation, expectations; type-domain notation
    - generative setting in terms of jointly i.i.d. random variables
    - convex losses, expected and conditional expected generalization loss
    - stylized theoretical results for common classes of supervised learners
    - formal learning task definitions, example: supervised classification
2. Introduction to object oriented programming (oop) and design patterns
    - Oop: classes and objects, structured types, related functional concepts
    - Object oriented programming in python, in R

    - Basic oop design patterns: template/strategy pattern, composite pattern
    - Running example: scikit-learn-like toolboxes and key oop patterns
    - UML diagrams, example designs, best design practice
3. Software engineering basics with applications to statistics and AI core logic
    - Model oriented design, domain driven design
    - Conceptual models and data models for scientific and mathematical scenarios
    - Example: supervised learning conceptual model, recap of mathematical objects
    - Common conceptual models and consensus design for common AI toolboxes: scikit-learn, weka, mlr
    - Selected advanced designs – toolboxes with computational or compiler-like back-ends (e.g., tensorflow)
4. Scientific types, algorithmic composition and reduction
    - Typing in programming languages
    - Basic type theory (first-order) from a formal perspective
    - Scientific types and mathematical types in toolboxes, relation to conceptual model
    - Example: random variables, probabilistic composites; supervised learning
    - Reduction and composition strategies in machine learning, including pipelines, tuning, and reduction with a change of task type
5. By participant preference.
   Option 1: hackathon with coding exercises or code contributions to open source tool, e.g., scikit-learn, mlr, sktime.
   Option 2: architecture for complex modelling tasks – time series, probabilistic prediction, event modelling
   Option 3: probabilistic programming and related software architectures

- Recommended reading:

Gamma, Erich, et al. *Elements of reusable object-oriented software*. Vol. 99. Reading, Massachusetts: Addison-Wesley, 1995.

Evans, Eric, and Eric J. Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.

James, Gareth, et al. *An introduction to statistical learning*. Vol. 112. New York: springer, 2013.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. No. 10. New York: Springer series in statistics, 2001.

Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* 12 (2011): 2825-2830.

Lang, Michel, et al. "mlr3: A modern object-oriented machine learning framework in R." *Journal of Open Source Software* 4.44 (2019): 1903.

Király, Franz J., et al. "Designing Machine Learning Toolboxes: Concepts, Principles and Patterns." *arXiv preprint arXiv:2101.04938* (2021).

- Prerequisites:

Basic knowledge of mathematical formalism

Beginner proficiency in python or R language

Prior exposure as user to data scientific toolboxes such as scikit-learn, mlr3

Format:

- 4 exercise sheets
- optionally: mentored contribution to selected open source project
- assessment: software implementation project (including design)
- Electronic lecture notes, available at end of course
- Necessary software requirements for computing facilities.
  One of the following, with admin user rights for package installation:
    - Computing environment with python>=3.6, python IDE (e.g., visual studio code), package environment manager (e.g., anaconda)
    - Computing environment with R Studio and R>=4.1.0
- Proposed timing: block 1, 1pm or 10:50am slot
- Lecture/computer session/tutorial/discussion split (hours of each): 2h lecture

Lecturer Details:

- Lecturer: Franz Kiraly
- Lecturer home institution: UCL
- Lecturer e-mail: f.kiraly@ucl.ac.uk
- Lecturer telephone number: N/A