Interoperability among Data Processing Frameworks: Reality or Wishful Thinking?

TIZIANO ZITO[a,b], NIKO WILBERT[c,a], RIKE-BENJAMIN SCHUPPNER[a],
ZBIGNIEW JĘDRZEJEWSKI-SZMEK[d], LAURENZ WISKOTT[c,a,e], PIETRO BERKES[f]

[a]*Bernstein Center for Computational Neuroscience, Berlin, Germany*
[b]*Modelling of Cognitive Processes, Berlin Institute of Technology, Germany*
[c]*Institute for Theoretical Biology, Humboldt-University, Berlin, Germany*
[d]*Institute of Experimental Physics, University of Warsaw, Poland*
[e]*Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany*
[f]*National Volen Center for Complex Systems, Brandeis University, Waltham, MA, USA*

May 23, 2011

Experimental and theoretical research in neuroscience constantly requires the application of data processing algorithms for data analysis and modeling. As Python is currently the second-most used programming language in the field, it is relatively easy to find implementations of the most common algorithms.

When building complete applications, scientists are often confronted with several additional chores that need to be carried out beside the individual processing steps. It is common to test different algorithms on the same data set (e.g., in control experiments), or to train and execute a sequence of several algorithms. When the algorithms come from different data processing libraries their implementations likely follow different conventions and do not share a common interface, making their interfacing cumbersome and error-prone.

In this talk, I will discuss the effort taken in the Modular toolkit for Data Processing (**`MDP`**) to ensure the interoperability with other mainstream data processing frameworks. **`MDP`** is a library that provides an implementation of several widespread algorithms, and offers a unified framework to combine them to build more complex data processing architectures. The general strategy to interface with other libraries has been to offer wrappers for the external algorithms. **`MDP`** currently features automatically generated wrappers for **`scikits.learn`**, another widely used data processing framework, and static wrappers for two Support Vector Machine libraries, **`shogun`** and **`libSVM`**. Additionally, **`MDP`** has been designed to be easy to embed in other frameworks, and I will show how this is achieved in **`PyMVPA`**, yet another data processing framework.

Using **`MDP`** as a paradigmatic example, I am going to explore current problems and challenges of achieving interoperability among data processing frameworks and draw some conclusions about the future.