RLPy: A Reinforcement Learning Framework for Education and Research

http://acl.mit.edu/rlpy

Alborz Geramifard†, Robert H. Klein†, Christoph Dann†, William Dabney*, Jonathan P. How†
† Laboratory for Information and Decision Systems, Massachusetts Institute of Technology
* University of Massachusetts Amherst

Many workflows in machine learning are based on processing data in discrete stages and storing the intermediate results on disk (e.g. many supervised learning scenarios), allowing flexibility in data processing methods. While such a pipeline works well on static, fixed-size datasets, it is not possible to modularize data processing in this way for sequential online decision-making problems such as reinforcement learning (RL) as RL learning algorithms (agents) affect the data collection phase through their actions. This property begs for a common protocol between the agent and the problem during the runtime. The RL-Glue project[1] took a first major step toward such a standard between components by defining a socket-based protocol that allows code from many programming languages to interact. However, the total generality of RL-Glue can be inconvenient in practice, limiting flexibility and hindering tools that might support an experiment throughout its lifetime. The RLPy framework promotes a different strategy. Instead of defining low-level protocol, it leverages the advantages of high-level, object-oriented programming in Python for managing the interaction between different components. This approach allows for easy modularization and reuse of code between components; for example, in RLPy the value function representation and policy are cast as separate classes and are easily exchangeable within a learning agent. Entirely new learning agents can be constructed from these smaller building blocks. This enhances and speeds exploratory research and education, but is much harder when assuming multiple component languages.

In RLPy all high-level tasks (such as controlling the experimental procedure, performance evaluation, and agent or domain prototyping) are performed in Python, which was selected for its user-friendliness, conciseness, and portability when compared with other languages. Requiring each component to have a Python interface may initially seem restrictive. However, low-level, time-critical RLPy tools are implemented in more efficient languages such as C++ and exposed through a Python wrapper, a common and successful practice in many scientific libraries such as *scikit-learn*[2] and *pandas*[3]. Additionally, recent projects such as *Cython* help to create wrappers around existing code with minimal effort. This allows RLPy to combine the speed of low-level languages and the ease of use of high-level languages such as easy debugging, code profiling (to identify bottlenecks), and interactive code execution.

RLPy provides tools for the entire experiment workflow, from initial conception to final plotting, and relies on existing open-source packages. For example, it allows easily and efficiently tuning hyperparameters of learning agents with the *hyperopt*[4] package. RLPy also includes tools for analyzing and producing paper-ready plots of experimental results based on the *matplotlib*[5] plotting library, but the user is not confined to the use of these tools. Intermediate information is stored on disk in common formats (e.g. JSON for the measured performance of algorithms) so that other tools can be used instead. For example, one might want to further analyze the results of a large experiment with the pandas data analysis library.

Fast prototyping and scalability to large problems are usually contradicting goals for a software system. Through low-level implementations of critical functions, natively parallelizable experiments, and a "batteries-included" approach via a large library of reusable components, RLPy aims to combine the ease of use of Python with practical scalability. Experiments can be run sequentially, in parallel on several CPU cores of a single machine or even on multiple machines in a job-based cluster system such as HTCondor.

RLPy includes interchangeable tools and components for each step of an RL experiment. These components can be combined with short Python scripts, which generate identical results on varying hardware. RLPy thus facilitates rapid and seamless experimental workflow from initial conception to final plotting, and helps researchers to provide small, reusable, and easily understandable packages along with their publications for reproducible research.

[1] http://glue.rl-community.org [2] http://scikit-learn.org [3] http://pandas.pydata.org
[4] http://jaberg.github.io/hyperopt [5] http://matplotlib.org