RLPy: A Value-Function-Based Reinforcement Learning Framework for Education and Research

Alborz Geramifard
Robert H. Klein
Christoph Dann
Machine Learning Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 – USA
William Dabney
University of Massachusetts Amherst, 140 Governors Dr., Amherst, MA 01003 – USA
Jonathan P. How
Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139 – USA

Abstract

RLPy is an object-oriented reinforcement learning framework with focus on value-function-based methods using linear function-approximation and discrete actions. The framework was designed for both, education and research purposes. It provides a rich library of fine-grained, easily exchangeable components for learning agents (e.g., policies or representations of value functions), facilitating recent increased specialization in reinforcement learning. RLPy is written in Python to allow fast prototyping but is also suitable for large-scale experiments by relying on optimized numerical libraries and parallelization. Code profiling, domain visualizations, and data analysis are integrated in a self-contained package available under the Modified BSD License at http://acl.mit.edu/rlpy. All these properties allow users to compare various RL algorithms with little effort.

Keywords: object-oriented reinforcement learning, empirical evaluation, open source

1. Introduction

An integral part of most introductory artificial intelligence courses are value-function-based methods using linear function approximation for solving Markov decision processes (such as linear Q-learning or SARSA). In addition, many researchers build upon this well-understood and powerful framework and aim at improving existing methods by, for example, feature learning (Geramifard et al., 2011; Keller et al., 2006; Parr et al., 2007) or policies that better trade off exploration and exploitation (e.g., Jaksch et al., 2010; Li, 2012; Nouri and Littman, 2009).

The need for unifying and reusable software packages for reinforcement learning research has been widely discussed (Schaul et al., 2010; Tanner and White, 2009), and many successful tools have been created (de Comite, 2006; Degris, 2013; Schaul et al., 2010; Tanner and White, 2009). However, it is desirable to have a software framework that is 1) easily accessible by novices so that they may compare and understand existing algorithms, and

©2014 Alborz Geramifard, Robert H. Klein, Christoph Dann, and Jonathan P. How.
efficient for researchers who desire to perform large scale experiments and advance the state-of-the-art.

By focusing on the prominent class of value-function-based methods with linear function approximation using discrete actions, RLPy aims at being such a software framework that provides simple and convenient tools for conducting sequential decision making experiments. In the following, we present the main features of RLPy and highlight the features that distinguish it from existing frameworks with different foci.

2. Existing Frameworks

The following existing software packages have some overlap with RLPy:

1. RL-Toolbox: (Neumann, 2005) C++ RL toolbox focusing on continuous state-spaces
2. CLSquare: (Riedmiller et al., 2012) C++ RL framework focused on interfaces with several robotics
3. libPG: (Aberdeen, 2007) RL library focused on high-performance policy-gradient algorithm implementations
4. rllib (Frezza-Buet and Geist, 2013): Template-based C++ RL library for value-function methods
5. rt-tezplot-ros-pkg (Hester, 2013) ROS package for RL algorithms
6. JRLF: (Kochenderfer, 2006) small-scale Java-Framework for RL experiments
7. PIQLE: (de Comite, 2006) Java-Framework for RL experiments
8. RLlib: (Abeyruwan, 2013) Port of RLPark into C++
10. JRLF: (Kochenderfer, 2006) small-scale Java-Framework for RL experiments
11. ApproxRL: (Busoniu, 2010) Matlab Toolbox with RL and dynamic programming algorithms
12. MMLF: (Metzen and Edgington, 2011) Python-based framework for reinforcement learning
13. PyBrain (Schaul et al., 2010): Machine learning library focused on neural networks with RL support

For the sake of brevity, we do not compare RLPy against each of the existing frameworks in detail but highlight key differences in the following section by referencing the list above.

3. Why RLPy?

Improved Granularity of Agents with Linear Value Functions. RL has advanced significantly over the past decade, leading researchers to narrow their focus towards specialized, independent aspects of RL agents, such as approximate function representations, exploration schemes, and learning rates. The structure of numerous existing frameworks (2, 6, 7, 10, 12) does not properly account for this increased specialization and makes it cumbersome to exchange, for example, the way the value function is represented in a learning agent. RLPy addresses this issue by separating these components into exchangeable classes (shown as green boxes in Figure 1) and other minor components such as learning rates into separate functions. This division reduces implementation effort, promotes reusability, and facilitates automated testing. Code for an example experiment that exploits this modularity is shown in Figure 2. In addition, the assumption of linearly parameterizing the value function allows RLPy to provide many tools and helpers for designing state features. For example, in large MDPs where using a tabular representation is infeasible, the IndependentDiscretization representation creates a set features by ignoring dependency among dimensions of the state space. Further dependency between such features can be added to the representation using the iFDD (incremental feature dependency discovery) module.

Rapid Prototyping with Python. RLPy is fully object-oriented and based primarily on the Python language (van Rossum and de Boer, 1991). Low-level, computationally-intensive tools are implemented in Cython (a compiled and typed version of Python) or C++. In contrast to other packages (1 – 9) written solely in C++ or Java, this approach leverages
Figure 1: RLPy framework - Green components constitute an RL agent which did not exist as separate components in previous RL frameworks. The experiment module handles the interaction between the agent and the domain; gray arrows depict the information flow in a conventional RL framework (See e.g. Sutton and Barto, 1998).

```python
import Agents, Representations, Policies, Experiments, Domains, Tools

logger = Tools.Logger()

### Domain ###
domain = Domains.InfCartPoleBalance(logger=logger)

### Agent ###
representation = Representations.Tabular(domain, logger, discretization=20)
policy = Policies.eGreedy(representation, logger, epsilon=0.1)
agent = Agents.SARSA(representation, policy, domain, logger)

### Experiment ###
experiment = Experiments.Experiment(agent, domain, logger, max_steps=100000)
experiment.run()
experiment.save()
```

Figure 2: RLPy code for setting up and running an experiment: SARSA learning for 100,000 steps how to balance an inverted pole on a cart while following an $\epsilon$-greedy policy. The representation is a discretized table lookup with 20 bins spaced uniformly in each dimension.

the user-friendliness, conciseness, and portability of Python while supplying computational efficiency where needed. This combination allows researchers to prototype new ideas quickly and comfortably without sacrificing the computing speed necessary to conduct large-scale experiments. In addition, the Python-based approach of RLPy is particularly suited for education as it does not require any proprietary software (in contrast to 11).

“Batteries Included” – Many Existing Components and Benchmarks. RLPy includes an ever-growing repository of components which may be combined to form new RL agents. While many frameworks (1, 3, 4, 6) only include classic benchmark domains such as PuddleWorld or an Inverted Pendulum on Cart, RLPy supplies a large number of more challenging domains such as HIV-Treatment, Hovering a Helicopter, and Pac-Man. In addition to implementations of most value-function-based RL algorithms, RLPy includes experimental support for dynamic programming methods that require full domain knowledge but yield optimal policies. This is especially useful as a baseline for comparison with (often sub-optimal) policies generated by RL agents.

Ease of Use and Development. Numerous tools are shipped with RLPy that facilitate ease of use and efficiency. One example is the code profiler, which produces a visual runtime graph of the source code (c.f. Figure 3 right) and identifies slow routines. This information allows the researcher to reduce the runtime of an algorithm with minimal effort and discourages premature runtime optimization. Additionally, every RLPy domain has a visualization, an important feature lacking in other frameworks (3, 4, 7). These visuals help
students and researchers quickly assess and gain intuition about the algorithm and domain behavior.

**Automation of Experiments** RLPy aims to promote reproducible research. To this end, it provides a suite of tools to automate the entire experiment pipeline. For example, RLPy allows concise specification of experiment settings (c.f. Figure 2) and automated and efficient hyperparameter optimization with the hyperopt package (Bergstra et al., 2013). Researchers can share their experimental setups by publishing short settings files, and colleagues can reproduce the results when running the scripts independent of their hardware or operating system. Additionally, RLPy experiments are natively parallelizable. Once parameters are selected, the user simply specifies the number of CPU cores RLPy can utilize for multiple experiments to test statistical significance. RLPy enables further scaling by switching seamlessly from a single machine to a job-based cluster (e.g. HTCondor) while ensuring results remain identical across varying hardware. RLPy also provides automated tools for generation of final publication-ready plots of results (c.f. Figure 3 left); researchers need only specify the quantities that should appear on the plot. To the best of our knowledge this degree of automation of the entire experimentation pipeline is unique to RLPy.

4. Conclusion

RLPy is a new reinforcement learning framework focused on value-function-based reinforcement learning using linear function approximation with discrete actions. It reflects recent developments in the field. It simplifies the construction of learning agents and makes it easier for novices and experts alike to evaluate and compare algorithms, representations, environments, and other RL components and modules, and to construct their own. It also provides many tools for conducting reproducible experiments from initial prototyping to final plotting. The framework is entirely open-source and all contributions are welcome and encouraged.

**Acknowledgments:** Many thanks to all those who helped develop and test RLPy in its early stages, especially David E. Williams. Thanks also to the ever-growing body of students and researchers who continue to improve and contribute to the project.
References


