Journal of Machine Learning Research 10 (2009) 2133-2136 Submitted 3/09; Published 9/09

RL-Glue: Language-Independent Software for Reinforcement-Learning Experiments

Brian Tanner BTANNER@CS.UALBERTA.CA
Adam White AWHITE@CS.UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8

Editor: Mikio L. Braun

Abstract

RL-Glue is a standard, language-independent software package for reinforcement-learning experiments. The standardization provided by RL-Glue facilitates code sharing and collaboration. Code sharing reduces the need to re-engineer tasks and experimental apparatus, both common barriers to comparatively evaluating new ideas in the context of the literature. Our software features a minimalist interface and works with several languages and computing platforms. RL-Glue compatibility can be extended to any programming language that supports network socket communication. RL-Glue has been used to teach classes, to run international competitions, and is currently used by several other open-source software and hardware projects.

Keywords: reinforcement learning, empirical evaluation, standardization, open source

1. Introduction and Motivation

Reinforcement learning is an embodied, trial-and-error problem formulation for artificial intelligence (Sutton and Barto, 1998; Kaelbling et al., 1996; Bertsekas and Tsitsiklis, 1996). At a series of time steps, the *agent* emits actions in response to observations and rewards generated by the *environment*. The agent's objective is to select actions that maximize the future rewards. Reinforcement-learning methods have been successfully applied to many problems including backgammon (Tesauro, 1994), elevator control (Crites and Barto, 1998), and helicopter control (Ng et al., 2004). Reinforcement-learning models and formalisms have influenced a number of fields, including operations research, cognitive science, optimal control, psychology, neuroscience, and others.

Reinforcement-learning practitioners create their agents and environments using various incompatible software frameworks, making collaboration inconvenient and thus slowing progress in our community. It can be time consuming, difficult, and sometimes even impossible to exactly reproduce the work of others. A conference or journal article is not the appropriate medium to share a sufficiently detailed specification of the environment, agent and overall experimental apparatus. We need a convenient way to share source code.

We believe that a standard programming interface for reinforcement-learning experiments will remove some barriers to collaboration and accelerate the pace of research in our field. To encourage widespread adoption, this interface should be easy to adhere to, and it should not force users to abandon their favorite tools or languages. With these goals in mind, we have developed RL-Glue: language independent software for reinforcement-learning experiments.

2. RL-Glue

Reinforcement-learning environments cannot be stored as fixed data-sets, as is common in conventional supervised machine learning. The environment generates observations and rewards in response to actions selected by the agent, making it more natural to think of the environment and agent as interactive *programs*. Sutton and Barto describe one prevalent view of agent-environment interactions in their introductory text (1998). Their view, shown in Figure 1, clearly separates the agent and environment into different components which interact in a particular way, following a particular sequence.

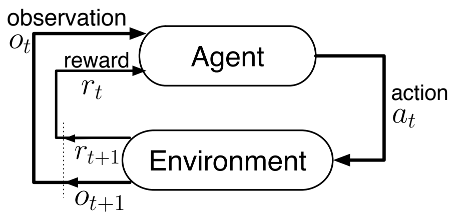

Figure 1: Sutton and Barto's agent-environment interface, with states generalized to observations.

White's RL-Glue Protocol (2006) formalizes Sutton and Barto's interface for online, single-agent reinforcement learning. The RL-Glue Protocol describes how the different aspects of a reinforcement-learning experiment should be arranged into programs, and the etiquette they should follow when communicating with each other. These programs (Figure 2) are the agent, the environment, the experiment, and RL-Glue. The agent program implements the learning algorithm and action-selection mechanism. The environment program implements the dynamics of the task and generates the observations and rewards. The experiment program directs the experiment's execution, including the sequence of agent-environment interactions and agent performance evaluation. The RL-Glue program mediates the communication between the agent and environment programs in response to commands from the experiment program. Our RL-Glue software (RL-Glue) implements White's protocol.[1]

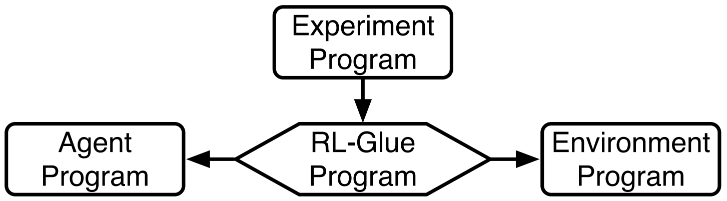

Figure 2: The four programs specified by the RL-Glue Protocol. Arrows indicate the direction of the flow of control.

RL-Glue can be used either in internal or external mode. In *internal* mode, the agent, environment and experiment are linked into a single program, and their communication is through function calls. Internal mode is currently an option if the agent, environment, and experiment are written exclusively in Java or C/C++. In *external* mode, the agent, environment and experiment are linked

1. This can be found at `http://glue.rl-community.org/protocol`.

into separate programs. Each program connects to the RL-Glue server program, and all communication is over TCP/IP socket connections. External mode allows these programs to be written in any programming language that supports socket communication. External mode is currently supported for C/C++, Java, Python, Lisp, and Matlab.

Each mode has its strengths and weaknesses. Internal mode has less overhead, so it can execute more steps per second. External mode is more flexible and portable. The performance difference between the two modes vanishes as the agent or environment becomes complex enough that computation dominates the socket overhead in terms of time per step. The agent and environment are indifferent and unaware of their execution mode; the difference in modes lies only in how the agent and environment are linked or loaded.

3. RL-Glue in Practice

RL-Glue's provides a common interface for a number of software and hardware projects in the reinforcement-learning community. For example, there is the annual RL-Competition, where teams from around the world compare their agents on a variety of challenging environments. The competition software uses the API, called RL-Viz, that is layered on top of RL-Glue to dynamically load agent and environment programs, modify parameters at runtime and visualize interaction and performance. All of the environments and sample agents created by the competition organizers are added to the RL-Library, a public, community-supported repository of RL-Glue compatible code. The RL-Library is also available as an archive of top competition agents, challenge problems, project code from academic publications, or any other RL-Glue compatible software that members of our community would like to share.

The socket architecture of RL-Glue allows diverse software and hardware platforms to be connected as RL-Glue environment programs. There are ongoing projects that connect a mobile robot platform, a keepaway soccer server, a real-time strategy game, and an Atari emulator to RL-Glue. Our socket architecture helps lower the barriers for researchers wishing to work on larger scale environments by providing a simple and familiar interface.

RL-Glue has been used for teaching reinforcement learning in several university courses and to create experiments for scientific articles published in leading conferences. See our *RL-Glue in practice* web page for an updated list of projects and papers that have used RL-Glue.[2]

4. Other Reinforcement-Learning Software Projects

RL-Glue is not the first software project that aims to standardize empirical reinforcement learning or to make agent and environment programs more accessible within our community. However, RL-Glue is the only project that offers a standardized language-independent interface, rich actions and observations, and fine-grained control of the experiment.

Other projects, most notably: CLSquare,[3] PIQLE,[4] RL Toolbox,[5] JRLF,[6] and LibPG,[7] offer significant value to the reinforcement-learning community by offering agents and environments,

2. This can be found at `http://glue.rl-community.org/rl-glue-in-practice`.
3. This can be found at `http://www.ni.uos.de/index.php?id=70`.
4. This can be found at `http://piqle.sourceforge.net/`.
5. This can be found at `http://www.igi.tugraz.at/ril-toolbox/`.
6. This can be found at `http://mykel.kochenderfer.com/jrlf/`.
7. This can be found at `http://code.google.com/p/libpgrl/`.

intuitive visualizations, programming tools, etc. Users should not be forced to choose between RL-Glue and these alternative projects. Our design makes it relatively easy to interface existing frameworks with RL-Glue. We are currently offering our assistance in bridging other frameworks to RL-Glue, with the hope of improving access to all of these tools for all members of our community.

5. RL-Glue Open Source Project

Website: http://glue.rl-community.org **License:** `Apache 2.0`

RL-Glue is more than an interface; it connects a family of community projects, with many levels of possible participation. Members of the community are invited to submit agent, environment and experiment programs to the RL-Library. Developers can also extend the reach of RL-Glue compatibility by writing external-mode or internal-mode interfaces for their favorite programming language. The RL-Glue software project also welcomes submissions and improvements for all parts of the software and documentation.

Acknowledgments

We would like to thank the users, testers, and developers for their contributions to RL-Glue 3.0. Special thanks to Gábor Balázs, José Antonio Martin H., Scott Livingston, Marc Bellemare, István Szita, Marc Lanctot, Anna Koop, Dan Lizotte, Richard Sutton, Monica Dinculescu, Jordan Frank, and Andrew Butcher. Of course, we also owe a great debt to all of the talented people responsible for the historic and ongoing development of RL-Glue.[8]

References

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996. ISBN 1886529108.

Robert H. Crites and Andrew G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Proceedings of the International Symposium on Experimental Robotics*, pages 363–372, 2004.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 1998.

Gerald Tesauro. TD-gammon, a self-teaching backgammon program achieves master-level play. *Neural Computation*, 6:215–219, 1994.

Adam White. A Standard System for Benchmarking in Reinforcement Learning. Master's thesis, University of Alberta, Alberta, Canada, 2006.

8. This can be found at `http://glue.rl-community.org/contributors-history`.