The Journal of Open Source Software

ReinforcementLearning: A Package to Perform Model-Free Reinforcement Learning in R

Nicolas Pröllochs[1, 2] **and Stefan Feuerriegel**[3]

1 University of Giessen **2** University of Oxford **3** ETH Zurich

DOI: 10.21105/joss.01087

Software

- Review
- Repository
- Archive

Submitted: 22 October 2018
Published: 12 June 2019

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC-BY).

Summary

Reinforcement learning refers to a group of methods from artificial intelligence where an agent performs learning through trial and error (Sutton & Barto, 1998). It differs from supervised learning, since reinforcement learning requires no explicit labels; instead, the agent interacts continuously with its environment. That is, the agent starts in a specific state and then performs an action, based on which it transitions to a new state and, depending on the outcome, receives a reward. Different strategies (e.g. Q-learning) have been proposed to maximize the overall reward, resulting in a so-called policy, which defines the best possible action in each state. As a main advantage, reinforcement learning is applicable to situations in which the dynamics of the environment are unknown or too complex to evaluate (e.g. Mnih et al., 2015). However, there is currently no package available for performing reinforcement learning in R. As a remedy, we introduce the `ReinforcementLearning` R package, which allows an agent to learn the optimal behavior based on sampling experience consisting of states, actions and rewards (Pröllochs & Feuerriegel, 2017). Based on such training examples, the package allows a reinforcement learning agent to learn an optimal policy that defines the best possible action in each state. Main features of `ReinforcementLearning` include, but are not limited to:

- Learning an optimal policy from a fixed set of a priori known transition samples
- Predefined learning rules and action selection modes
- A highly customizable framework for model-free reinforcement learning tasks

Statement of Need

Reinforcement learning techniques can primarily be categorized in two groups, namely, model-based and model-free approaches (Sutton & Barto, 1998). The former, *model-based* algorithms, rely on explicit models of the environment that fully describe the probabilities of state transitions, as well as the consequences of actions and the associated rewards. Specifically, corresponding algorithms are built upon explicit representations of the environment given in the form of Markov decision processes. These MDPs can be solved by various, well-known algorithms, including value iteration and policy iteration, in order to derive the optimal behavior of an agent. These algorithms are also implemented within the statistical software R. For instance, the package `MDPtoolbox` solves such models based on an explicit formalization of the MDP, i.e. settings in which the transition probabilities and reward functions are known a priori (Chades, Chapron, Cros, Garcia, & Sabbadin, 2017).

The second category of reinforcement learning comprises *model-free* approaches that forgo any explicit knowledge regarding the dynamics of the environment. These approaches learn the optimal behavior through trial-and-error by directly interacting with the environment. In

this context, the learner has no explicit knowledge of either the reward function or the state transition function (Sutton & Barto, 1998). Instead, the optimal behavior is iteratively inferred from the consequences of actions on the basis of past experience. As a main advantage, this method is applicable to situations in which the dynamics of the environment are unknown or too complex to evaluate. However, the available tools in R are not yet living up to the needs of users in such cases. In fact, there is currently no package available that allows one to perform model-free reinforcement learning in R. Hence, users that aim to teach optimal behavior through trial-and-error learning must implement corresponding learning algorithms in a manual way.

As a remedy, we introduce the `ReinforcementLearning` package for R, which allows an agent to learn the optimal behavior based on sampling experience consisting of states, actions and rewards (Pröllochs & Feuerriegel, 2017). The training examples for reinforcement learning can originate from real-world applications, such as sensor data. In addition, the package is shipped with the built-in capability to sample experience from a function that defines the dynamics of the environment. In both cases, the result of the learning process is a highly interpretable reinforcement learning policy that defines the best possible action in each state. The package provides a remarkably flexible framework, which makes it readily applicable to a wide range of different problems. Among other functions, it allows one to customize a variety of learning parameters and elaborates on how to mitigate common performance in common solution algorithms (e.g. experience replay). The package vignette demonstrates its use by drawing upon common examples from the literature (e.g. finding optimal game strategies).

Functionality

The `ReinforcementLearning` package utilizes different mechanisms for reinforcement learning, including Q-learning and experience replay. It thereby learns an optimal policy based on past experience in the form of sample sequences consisting of states, actions and rewards. Consequently, each training example consists of a state-transition tuple $(s_i, a_i, r_{i+1}, s_{i+1})$ as follows:

- s_i is the current environment state.
- a_i denotes the selected action in the current state.
- r_{i+1} specifies the immediate reward received after transitioning from the current state to the next state.
- s_{i+1} refers to the next environment state.

The training examples for reinforcement learning can (1) be collected from an external source and inserted into a tabular data structure, or (2) generated dynamically by querying a function that defines the behavior of the environment. In both cases, the corresponding input must follow the same tuple structure $(s_i, a_i, r_{i+1}, s_{i+1})$.

Learning from pre-defined observations is beneficial when the input data is pre-determined or one wants to train an agent that replicates past behavior. In this case, one merely needs to insert a tabular data structure with past observations into the package. This might be the case when the state-transition tuples have been collected from an external source, such as sensor data, and one wants to learn an agent by eliminating further interaction with the environment.

An alternative strategy is to define a function that mimics the behavior of the environment. One can then learn an agent that samples experience from this function. Here the environment function takes a state-action pair as input. It then returns a list containing the name of the next state and the reward. In this case, one can also utilize R to access external data sources, such as sensors, and execute actions via common interfaces. The structure of such a function is represented by the following pseudocode:

```
environment <- function(state, action) {
  ...
  return(list("NextState" = newState,
              "Reward" = reward))
}
```

After specifying the environment function, one can use `sampleExperience()` to collect random sequences from it. Thereby, the input specifies number of samples (N), the environment function, the set of states (i.e. S) and the set of actions (i.e. A). The return value is then a data frame containing the experienced state transition tuples $(s_i, a_i, r_{i+1}, s_{i+1})$ for $i = 1, \ldots, N$.

Notes on performance

Q-learning is guaranteed to converge to an optimal policy. However, the method is computationally demanding as it relies on continuous interactions between an agent and its environment. To remedy this, the `ReinforcementLearning` package allows users to perform batch reinforcement learning. In most scenarios, this reinforcement learning variant benefits computational performance as it mitigates the 'exploration overhead' problem in pure online learning. In combination with experience replay, it speeds up convergence by collecting and replaying observed state transitions repeatedly to the agent as if they were new observations collected while interacting with the system. Nonetheless, due to the fact that the package is written purely in R, the applicability of the package to very large scale problems (such as applications from computer vision) is still limited. In the following, we briefly summarize scenarios the package is capable of handling and situations in which one should consider utilizing reinforcement learning implementations written in "faster" programming languages.

What the `ReinforcementLearning` R package can do:

- Learning optimal strategies for real-world problems with limited state and action sets (e.g. finding optimal strategies for simple games, training a simple stock market trading agent, learning polarity labels in applications from natural language processing).
- The packages allows one to speed up performance by adjusting learning parameters and making use of experience replay.
- The package allows one to train an agent from pre-defined observations without the need for modeling the dynamics of the environment. Typically, this approach drastically speeds up convergence and can be useful in situations in which the state-transition tuples have been collected from an external source, such as sensor data.
- The package provides a highly customizable framework for model-free reinforcement learning tasks in which the functionality can easily be extended. For example, users may attempt to speed up performance by defining alternative reinforcement learning algorithms and integrating them into the package code.

What the `ReinforcementLearning` R package cannot do:

- Solving large-scale problems with high-dimensional state-action spaces such as those from computer vision (users may consider reinforcement learning implementations written in "faster" programming languages)
- Solving reinforcement learning problems requiring real-time interaction (e.g. real-time interaction with a robot)

References

Chades, I., Chapron, G., Cros, M.-J., Garcia, F., & Sabbadin, R. (2017). *MDPtoolbox*. doi:10.1111/ecog.00888

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi:10.1038/nature14236

Pröllochs, N., & Feuerriegel, S. (2017). *ReinforcementLearning*. Retrieved from https://CRAN.R-project.org/package=ReinforcementLearning

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Adaptive computation and machine learning. Cambridge, MA: MIT Press. doi:10.1109/TNN.1998.712192