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

22 October 2018

Summary

Reinforcement learning refers to a group of methods from artificial intelligence where an agent performs learning through trial and error (Sutton and 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, 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 and 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.

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 $i = 1, \ldots, N$.

References

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. "Human-Level Control Through Deep Reinforcement Learning." *Nature* 518 (7540): 529–33.

Pröllochs, Nicolas, and Stefan Feuerriegel. 2017. *ReinforcementLearning*. https://CRAN.R-project.org/package=ReinforcementLearning.

Sutton, Richard S., and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press.