RLCard: A Toolkit for Reinforcement Learning in Card Games

Daochen Zha†, Kwei-Herng Lai†, Yuanpu Cao†, Songyi Huang‡, Ruzhe Wei†, Junyu Guo†, Xia Hu†

†*Department of Computer Science and Engineering, Texas A&M University*
‡*Simon Fraser University, BC, Canada*

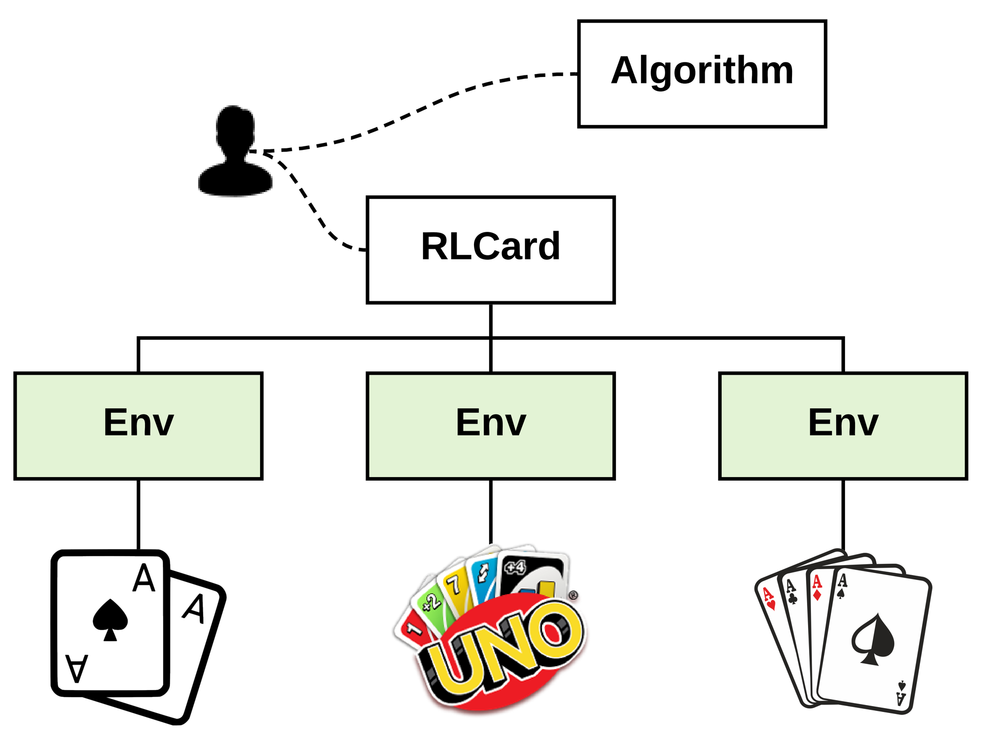
Motivations

- We have made great achievements in solving challenging imperfect information games, whereas there are few open-source libraries to help us develop and test our algorithms.
- Researchers, especially new researchers, usually need to spend huge engineering efforts to develop game engines.
- A large number of efforts of reinforcement learning have been made on the single-agent setting.
- We are motivated to develop an easy-to-use open-source library to bridge reinforcement learning and imperfect information games to enable more people to study imperfect information games.

Overview

We implement and wrap card games with easy-to-use unified interfaces. Our goal is to enable people to focus on algorithm design instead of the engineering efforts of developing games. The design follows the following principles:

- **Reproducible.** Results on the environments can be reproduced. The same result should be obtained with the same random seed in different runs.
- **Accessible.** Experiences are collected and well organized after each game with straightforward interfaces. State representation, action encoding, reward design, or even the game rules, can be conveniently configured.
- **Scalable.** New card environments can be easily added to the toolkit. We try to minimize the dependencies in the toolkit so that the codes can be easily maintained.



Available Environments

Environment	InfoSets	InfoSize	ActionSize
Blackjack	10^3	10^1	10^0
Leduc Hold'em	10^2	10^2	10^0
Limit Texas Hold'em	10^{14}	10^3	10^0
Dou Dizhu	$10^{53} \sim 10^{83}$	10^{23}	10^4
Mahjong	10^{121}	10^{48}	10^2
No-limit Texas Hold'em	10^{162}	10^3	10^4
UNO	10^{163}	10^{10}	10^1

We provide approximation of the difficulties considering the following aspects. **InfoSets:** the number of the information sets; **InforSize:** the average number of states in a single information set; **ActionSize:** the size of the action space (without abstraction).

Why Dou Dizhu and Mahjong?

We introduce two Chinese Poker games, which pose new challenges.

- **Large Action space**: Games such as Dou Dizhu support complex combinations of cards, which lead to more than 10^4 possible legal actions.
- **Large size of information set**: A large fraction of information is hidden from the player.
- **Sparse reward**: The rewards are very sparse. Only one winning hand is observed in Mahjong out of 500 random games.
- **Cooperation**: In Dou Dizhu, the two peasants have to work together to fight against the landlord.

To study these challenges, we also provide smaller versions of Dou Dizhu and Mahjong to capture the key dynamics of the games.

Interfaces

- **Basic interface**: We provide a `run` interface to generate a complete game data. This interface is easy-to-use for sampling-based algorithms such as DQN and NFSP.
- **Advanced interface**: We provide `step` to move to the next state and `step_back` to go back to the previous state. This interface is designed for tree-based algorithms such as CFR.

Evaluation

Exploitability is usually used for small games. However, for large games like Dou Dizhu, computing exploitability itself is challenging. Here, we provide the results of tournaments. Note that we only apply CFR on Leduc Hold'em, since the other games are too large for CFR.

Tournament	NFSP	DQN
Leduc Hold'em	1.0691	-1.0691
Limit Texas Hold'em	-0.0308	0.0308
Dou Dizhu with NFSP landlord	0.7049	0.2951
Dou Dizhu with DQN landlord	0.7303	0.2697
Mahjong	-0.0090	-0.0104
No-limit Texas Hold'em	9.5610	-9.5610
UNO	-0.0428	0.0428
	CFR	**NFSP**
Leduc Hold'em	0.0776	-0.0776
	CFR	**DQN**
Leduc Hold'em	1.2493	-1.2493

Ongoing Efforts and Future Work

We wish to do our best to serve the community. Some new features are under development and should be available soon.

- **Graphical Interfaces**: We are developing GUI for games to support easy model debugging.
- **Leaderboard**: A leaderboard will be developed to compare algorithms with tournaments.
- **Tree-based Wrapper**: We are supporting tree-based interfaces, i.e., operating on the tree in a style of parent and child nodes.

RLCard is available at `https://github.com/datamllab/rlcard` or the QR code below. We appreciate any generous comments.