Exploiting opponent's strategy in Poker

CS 594 - Reinforcement Learning
Final Project

University of Illinois at Chicago

Giuseppe Cerruto (gcerru2@uic.edu)
Edoardo Stoppa (estopp2@uic.edu)

Introduction

We wanted to change the focus from ***mastering*** a game to ***exploiting*** an opponent's strategy

What did we investigate in this project?

- Agents performance in multiplayer games
- Performance of a "naive" way of doing self-play

Algorithms used

Deep Q Network
(***DQN***)

DQN with prioritized
experience replay
(***PDQN***)*

***Implemented by us and integrated in the library (RLCard)**

Neural Fictitious Self-Play
(***NFSP***)

- One network learns best response using average adversary behaviour
- One network learns the the strategy through reinforcement learning

Card Games

Leduc Hold’em (simple version)

- Deck of 6 cards
- Each player is dealt a card privately
- A single card is dealt face up on the table
- Players reveal their card, if any player’s private card has the same rank as the one on the table wins, otherwise the person with the highest valued card wins

Limit Hold’em

The general poker game structure applies, but with some extremely important caveats:

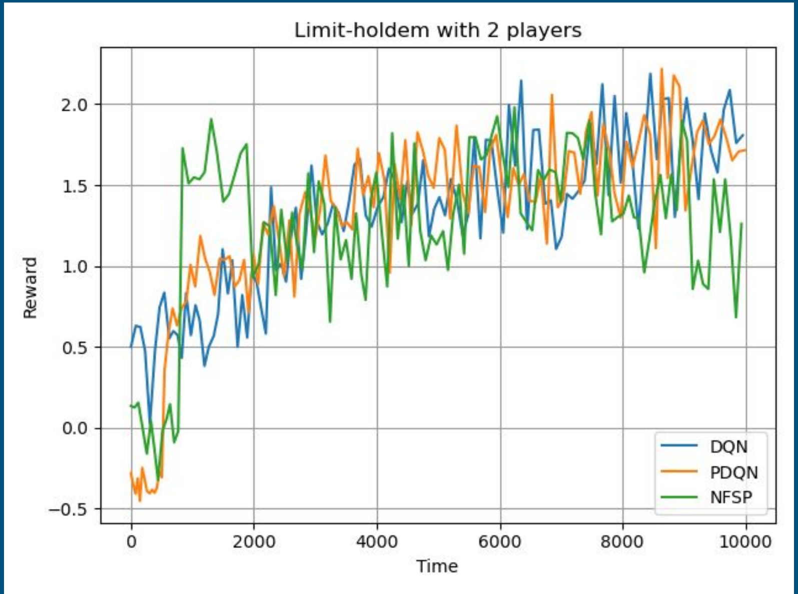
- Amount of money you can bet is limited (in a small range with fixed increments)
- In a single betting round, at most only 1 bet and 3 raises are allowed (after that everyone can only call or fold)

EXPERIMENTS

Agents Training

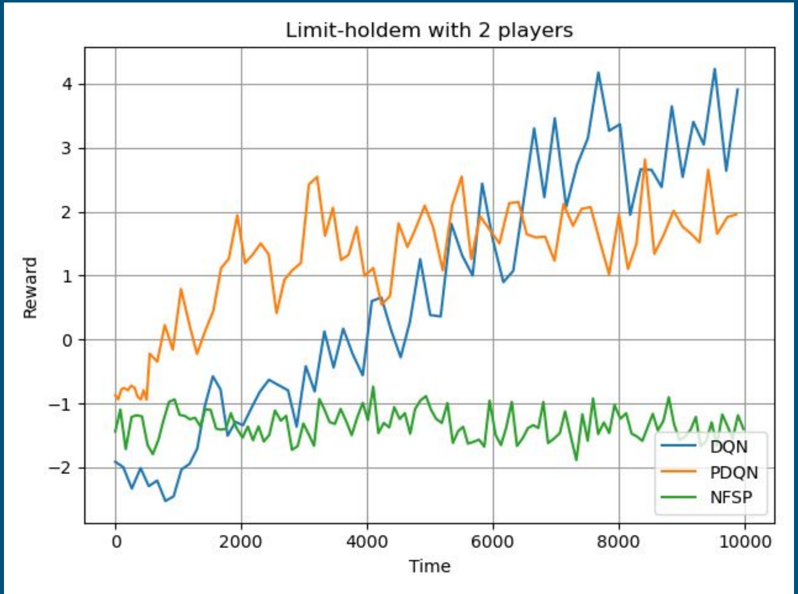
Baseline

(trained vs RandomAgent)



Expert

(trained vs Baseline)



Agent performance with an increasing number of players

Baseline vs Random

	3P	4P	5P	6P
DQN	-1.49%	-3.15%	-2.97%	-3.07%
PDQN	+1.88%	-0.87%	+0.24%	-0.29%
NFSP	+1.17%	+3.22%	-4.48%	-1.09%

Expert vs Baseline

	3P	4P	5P	6P
DQN	-1.87%	-0.02%	-0.37%	-0.01%
PDQN	-0.41%	+0.69%	-0.30%	-0.07%
NFSP	-2.41%	+3.72%	-0.71%	-2.58%

- Very minor change in performance (always within ±4%)
- DQN was the most consistent, while PDQN and NFSP showed slightly more erratic performance

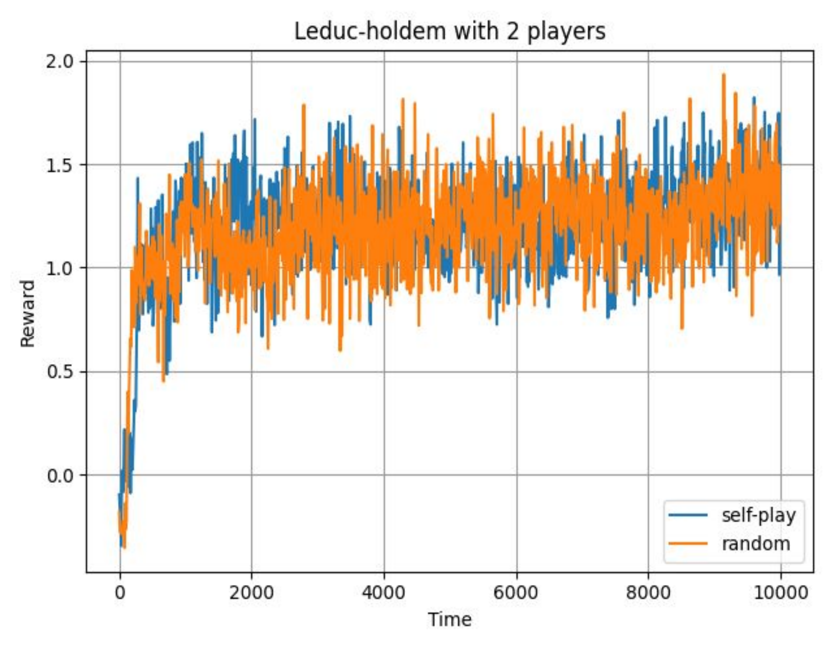
Agent performance with an increasing number of players

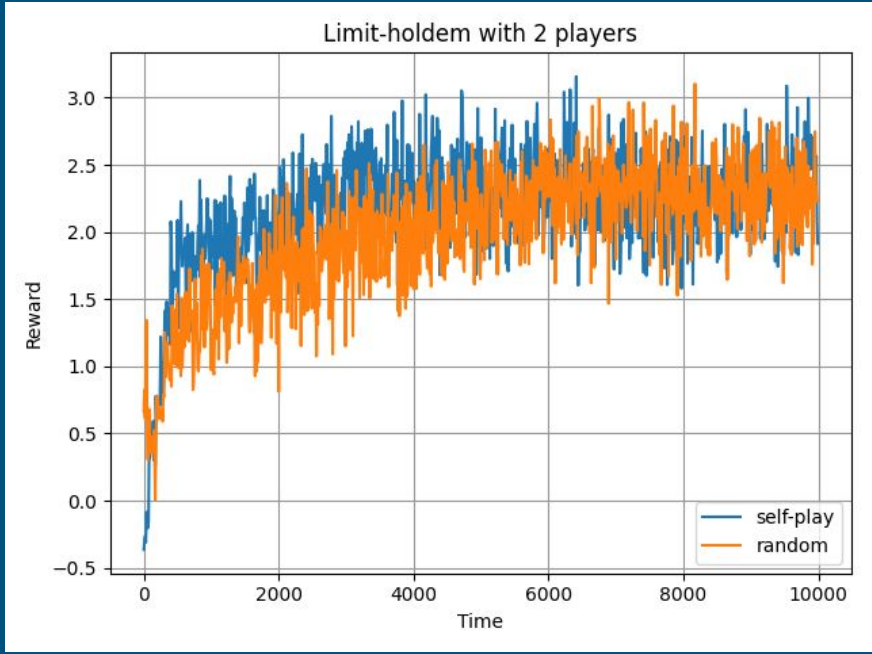
Expert vs Random

	3P	4P	5P	6P
DQN	-3.65%	-10.54%	12.66%	0.28%
PDQN	0.55%	10.31%	-6.76%	3.49%
NFSP	26.96%	-19.3%	-6.96%	-14.03

- Completely unpredictable results
- Significative variance in average reward

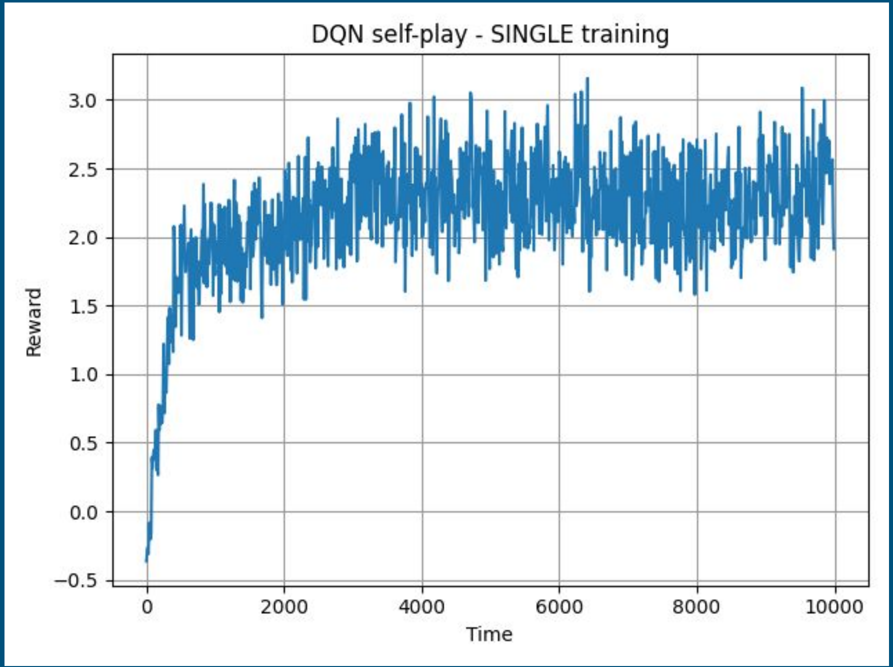
Agent performance: training against random agent VS “naive” self-play

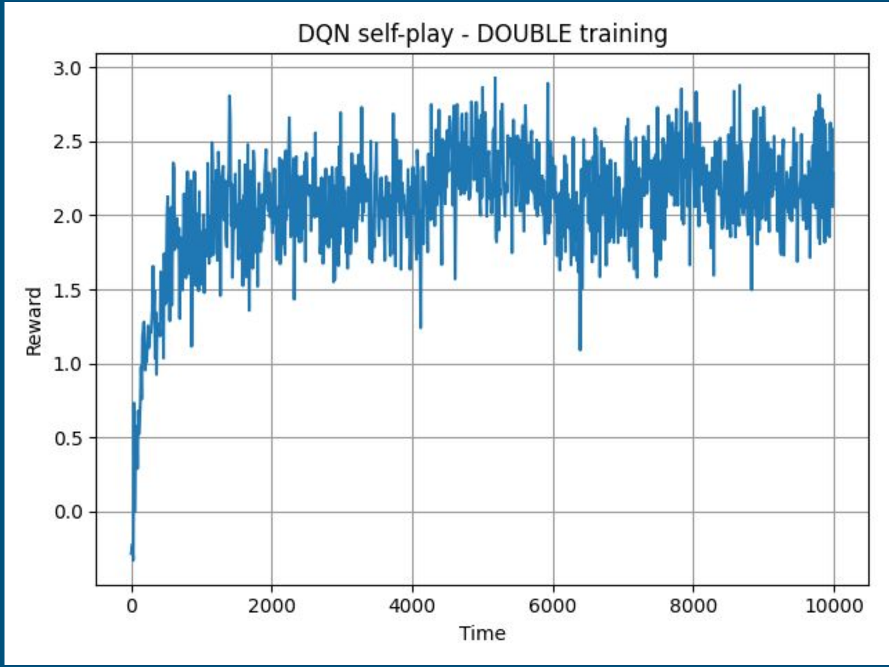




- Game is too simple to have significant improvements
- Faster convergence in the self-play mode

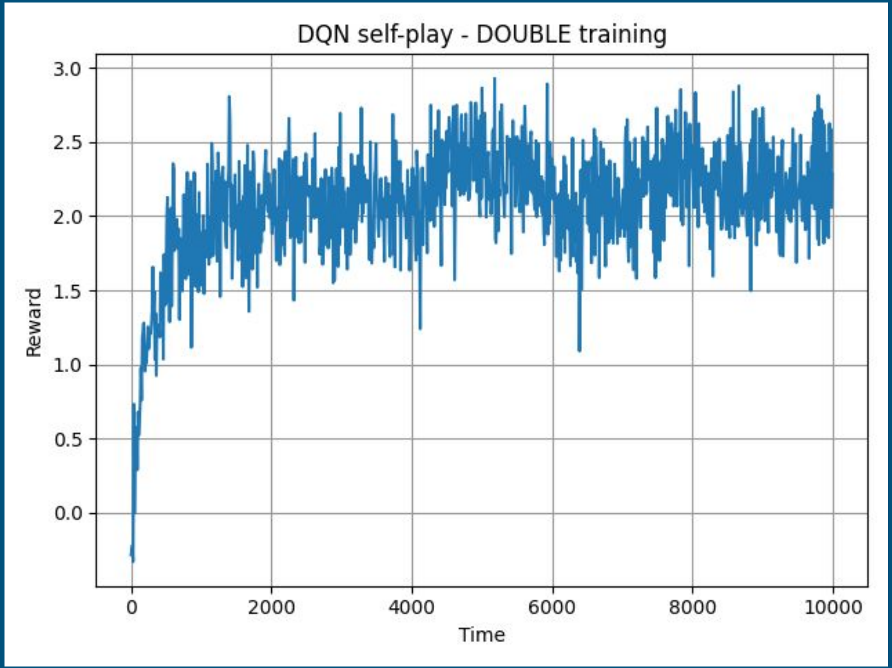
Agent performance: “naive” self-play - single vs double training

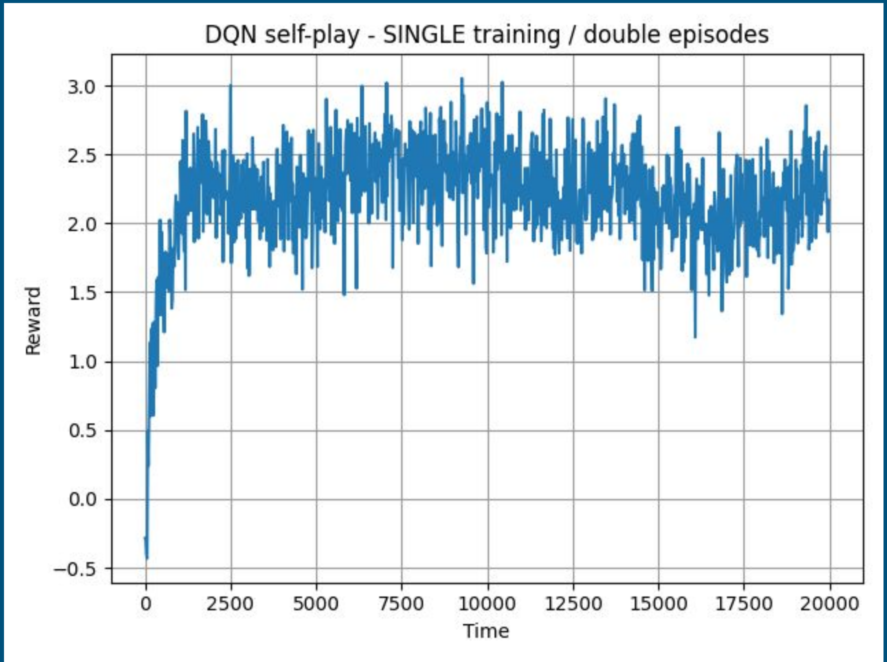




- No particular improvement in the double-training mode and instead more instability

Agent performance: “naive” self-play - double training vs single training/ double episodes



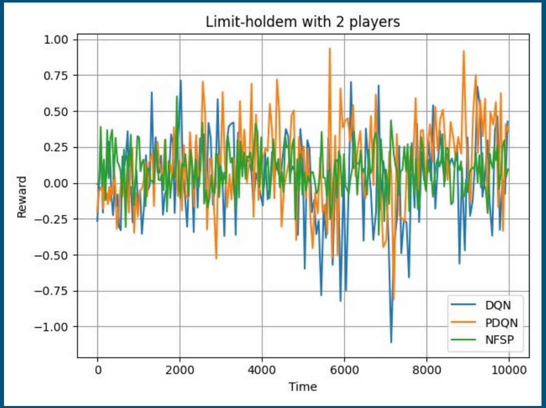


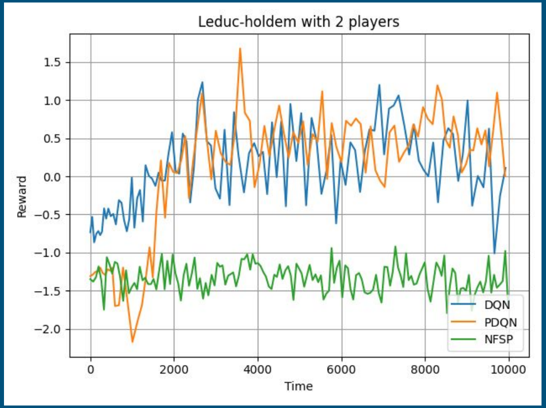
- Having the same number of transactions: single-training with double-episodes is preferred

Other experiments

- Training against hard-coded expert (already in the library)
- Tournament among experts
- etc…

GiuseppeCerruto/Exploiting_Poker (github.com)





Thanks for your attention!

Questions?

Giuseppe Cerruto (gcerru2@uic.edu)
Edoardo Stoppa (estopp2@uic.edu)

Github:https://github.com/GiuseppeCerruto/Exploiting_Poker