Applied Deep Learning Project - Initiate

Magnus Wagner (12034922)

October 25, 2021

1 Project description

For my project in Applied Deep Learning, I will try to build a Deep Reinforcement Learner that is able to play the popular card game *Wizard*. Wizard is a great example to simulate as the complexity rises with the number of players and the number of cards to be played. The number of cards increases each round, with only one card being played in the first round, two in the second round and so on. On top of that, there are two types of special cards and a betting round before the actual game which increases the complexity even more. The complete rule set can be found inside of the project repository. For the project, those complexity options can be enabled and model input can be adapted as soon as the previous complexity level was mastered by the learner. To start simple, the betting round is not part of the simulation at first and the number of cards to be played is low with only two players playing.

2 Dataset

As a simulation environment, I will try to adapt the RLCard repository to simulate the environment for Wizard.[1] [3] If it shows that this creates more problems than it solves, I will build the environment on my own in a less verbose and graphical way without any imported classes and constructs. The creation of the simulation environment would be equal to bringing my own data as the training data is created while running the simulation.

3 Method

For the learner, a Neural Fictitious Self-Play (NFSP) agent which is a deep reinforcement learner focused on self-play in imperfect information games will be implemented by me.[1] As the information for each player is limited in Wizard, this should work just as well as the application of NFSP in poker games. There is already an existing implementation for NFSP in the RLCard repository which can be adapted at first to test if the environment is working. This will include the adaptation of the input as much more information is available in a game of Wizard in comparison to a game of Limit Hold'Em Poker. If the implementation works and can be trained via self-play, this would already solve the part of the assignment regarding "Bring your own dataset." To improve upon the already implemented NFSP agent from RLCard, the algorithm can be improved via Monte Carlo Tree Search which is also sensible for a game of Wizard as only the last trick of a round decides if the total reward is positive or negative.[5][4] If this part of the assignment works, it would fulfill the assignment type "Bring your own method." General concepts of Reinforcement Learning are derived from the book "Reinforcement Learning - An Introduction" by Barto & Sutton[2] which was the main knowledge source of the TU course "AKNUM Reinforcement Learning" by Prof. Heitzinger.

4 Project Plan

The building of the environment is planned to happen during the first 30 hours of the programming part for this course. This time estimation includes the adaptation of the pre-implemented NFSP agent from RLCard and self-play training to generate an intelligent agent (approximately 10 hours).

[1] https://github.com/datamllab/rlcard

Regarding the course of the semester, I should finish this part until mid December. If everything works well with the pre-implemented NFSP agent after that time, I will try to spend the last 15 hours to implement the Monte Carlo Tree Search adapted version of the NFSP agent and train it again. The preparation of the final report and the presentation should take 10h and 4h respectively to match the expected hours in the TISS description of the course. As the project is not that simple and the training could potentially not deliver promising results at first, the time estimations might be set too low. This would result in me only focusing on building the environment and training the model via self-play.

References

[1] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[3] Daochen Zha, Kwei-Herng Lai, Yuanpu Cao, Songyi Huang, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019.

[4] Li Zhang, Yuxuan Chen, Wei Wang, Ziliang Han, Shijian Li, Zhijie Pan, and Gang Pan. A monte carlo neural fictitious self-play approach to approximate nash equilibrium in imperfect-information dynamic games. *Frontiers of Computer Science*, 15(5):1–14, 2021.

[5] Li Zhang, Wei Wang, Shijian Li, and Gang Pan. Monte carlo neural fictitious self-play: Approach to approximate nash equilibrium of imperfect-information games. *arXiv preprint arXiv:1903.09569*, 2019.