P05 – Selfplay for Tic Tac Toe

1. Background research

Amongst several striking ideas, Peter Abbeel prominently mentions "self play" as one important ingredient of future successful AI agents trained with reinforcement learning (RL) in his recent summary talk at EECS [1], where he places this topic in larger context of "learning to learn".

Make yourself acquainted with Prof. Abbeel's ideas from the abovementioned talk on "learning to learn" [4] with a specific focus on how the "self play" he proposes [2][3] works. Compare it with the work of David Silver et al. on the game of Go [5].

2. Learning to play Tic Tac Toe, WarGames-style

In their introductory book to RL, Sutton & Barto relate the example of the game Tic Tac Toe that due to its simplicity can be taught to an agent with a very simple RL technique [6, p. 7-10]. Please implement this example in a way where the agent learns to play Tic Tac Toe by repeated games against itself (i.e., against a non-static opponent that itself improves over time). Please use Python to implement the challenge and fulfill the API for the agent that is implicitly defined in `t3_controller.py` and `t3_engine.py`.

Check the progress of your agent while letting it play against humans (as first- and second mover). What do you observe, and why? Note down lessons learned from your development experience. Do you come to similar conclusions as the author of [7]?

Details

In the zip file, you can find two Python files:

- `t3_controller.py`: this is the main program, which you should use/extend
- `t3_engine.py`: this contains the game engine / helper functions for the Tic Tac Toe game, in particular it makes a move, evaluates the board (win/lose/draw), initializes board, prints board, etc.

Your task is to write a new, third python file called `t3_agent.py`

- You can then import this file from `t3_controller.py`
  - `import t3_agent as agent`
- In `t3_agent.py`, you need to provide at least the following methods
  - `model=agent.Load_model(file_name)`
  - `field = agent.get_best_action(board, model, turn)`

Besides providing at least the two methods above, you need to implement the training code in `t3_controller.py` (the code within the "`if training_mode:`" block starting at line 27). The idea is the following

- Create an outer loop of training for `N` games
- For each game
  - Start with the initial board
  - X and O make alternating moves (self-play)
  - Provide a function of the agent, which makes the move and learns from it, i.e.
    - choosing greedily the best move (or worst, depending on who plays) according to the current model with probability (`1-exploration_rate`), or
    - chosing a random move with probability `exploration_rate`
    - updating the board (from current state to new state `s'`) with the chosen move
    - updating the model with the temporal difference learning updating rule: `V[s] <- V[s] + step_size * (V[s']-V[s])`, where `V` is the value function
  - When the board is finished, determine who wins
  - The agent learns from the final move
    - Provide a function of the agent, which learns from the final move. To profit from the final state of the game (possibly reached by a winning move of the opponent), this function updates the last state of the losing player with the information that he is going to lose.

The `model` (==(going to be) trained RL agent) consists of a dictionary `V` that maps states (=board configurations, see description in `t3_engine.py`) to values (estimated probability to win from this state for the first player, X).

The learning method to implement is discussed in the book by Sutton & Barto [6], section 1.5, pages 7-10 ("temporal difference learning", equation on page 8)

References

[1] Peter Abbeel, "Learning to Learn Robotic Control", EECS colloquium, Oct 11, 2017, available online: https://www.youtube.com/watch?v=TERCdog1ddE&t=3575s (Feb 19, 2018)
[2] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, Pieter Abbeel, "Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments", Oct 10, 2017, available online: https://arxiv.org/abs/1710.03641 (Feb 19, 2018)
[3] Trapit Bansal, Igor Mordatch, Jakub Pachocki, Ilya Sutskever, Szymon Sidor, "Competitive Self-Play", OpenAI blog, Oct 11, 2017, available online: https://blog.openai.com/competitive-self-play/ (Feb 19, 2018)

[4] Chelsea Finn, "Learning to learn", BAIR blog, Jul 18, 2017, available onlinbe: http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/ (Feb 19, 2018)
[5] David Silver, Julian Schrittwieser, Karen Simonyan, et al., “Mastering the game of Go without human knowledge”, Nature, 550(7676), 354, 2017, available online: https://www.nature.com/articles/nature24270 (Sep 04, 2018)
[6] Richard S. Sutton, Andrew G. Barto, "Reinforcement Learning, 2nd Edition", MIT press (to appear), available online: http://incompleteideas.net/book/bookdraft2017nov5.pdf (Feb 19, 2018)
[7] Matthew Rahtz, "Lessons Learned Reproducing a Deep Reinforcement Learning Paper", Amid Fish Blog, April 06, 2018, available online: http://amid.fish/reproducing-deep-rl (Jul 17, 2018)