TicTacToe Environment

Check “TicTacToe/tictactoe/tictactoe/envs/tictactoe_env.py” for the implementation

Player symbol: X , Opponent symbol: O

Implementation internally handles the moves played by the opponent. One can set the opponent type as random, safe, any(randomly chosen between safe or random at runtime) or custom.

For custom opponent we need to pass the opponent policy function as well. One can set the opponent using set_opponent() method.

If any player makes invalid move then the other player automatically wins the game.

A state represents one of the possible tictactoe configuration. Cells of tictactoe board is numbered from 0 to 8. A state is tuple of 9 values each value can be 1 or -1 representing X or O respectively as mark on the corresponding cell.

Action is marking on one of the cells. Using step() method player makes one move and internally the opponent makes one move after which step() method returns the next state, reward and other information.

Rewards:

- Invalid move by player: status: lose, reward = -5
- Winning move by player: status: win, reward = 2
- Draw move by player: status : draw, reward = 1
- Blocking move by player: status: play, reward = 1
- Any other move by player: status: play, reward = 0

- Invalid move by opponent : status: win, reward +=0
- Opponent wins: status: lose, reward += -2
- Draw after opponent move: status: draw, reward += 1
- Any other move by opponent: status: play, reward +=0

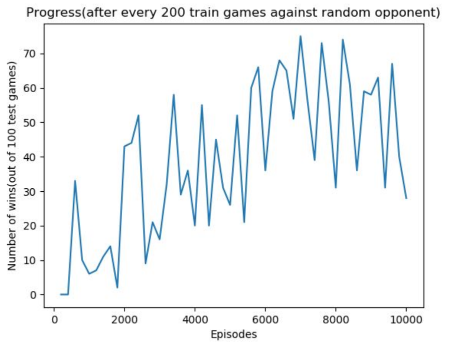
Final reward is received after both player and opponent make one move.

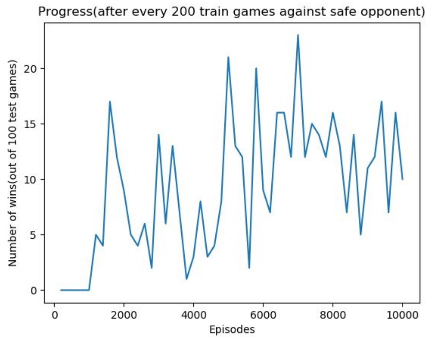
Mainly reward is received at the end of the episode depending on the status(lose:-ve reward, win: high +ve reward, draw: low +ve reward), except the case when player make blocking move which give +ve reward of 1.

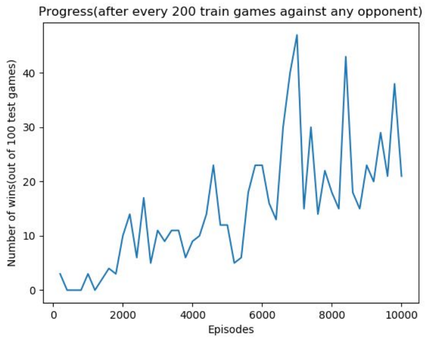
In output files one can see the TicTacToe game status and renderings during test games.

Training and Evaluation

Progress Plots







Below are the results of 3 tictactoe agents trained(10k games) against random, safe, any opponent respectively and tested against random, safe and self as custom opponent.

It looks agent trained against random opponent performed best among other agents based on the number of wins during testing.

It was observed that each of these agents can be improved further by training(1k games or more) against itself as a custom opponent .

Test results:

Agent trained against	Test results(Random)	Test results (Safe)	Test Results(self)
random	avg test reward: -0.855 {'win': 323, 'draw': 145, 'lose': 532}	avg test reward: -1.363 {'win': 114, 'draw': 220, 'lose': 666}	avg test reward:-0.031 {'win': 947, 'draw': 0, 'lose': 53}
safe	avg test reward:-3.605 {'win': 17, 'draw': 99, 'lose': 884}	avg test reward:-0.328 {'win': 107, 'draw': 364, 'lose': 529}	avg test reward:-0.146 {'win': 929, 'draw': 0, 'lose': 71}
any	avg test reward:-1.63 {'win': 248, 'draw': 109, 'lose': 643}	avg test reward:-0.958 {'win': 100, 'draw': 344, 'lose': 556}	avg test reward:0.023 {'win': 945, 'draw': 0, 'lose': 55}

Improved results after further training self as custom opponent:

Agent trained against	Test results(Random)	Test results (Safe)	Test Results(self)
random	avg test reward: 0.506 {'win': 626, 'draw': 53, 'lose': 321}	avg test reward: -0.023 {'win': 127, 'draw': 276, 'lose': 597}	avg test reward:-0.137 {'win': 942, 'draw': 0, 'lose': 58}
safe	avg test reward::-3.34 {'win': 53, 'draw': 112, 'lose': 835}	avg test reward:0.738 {'win': 170, 'draw': 536, 'lose': 294}	avg test reward:-0.073 {'win': 944, 'draw': 0, 'lose': 56}
any	avg test reward:-0.112 {'win': 525, 'draw': 113, 'lose': 362}	avg test reward:0.014 {'win': 98, 'draw': 545, 'lose': 357}	avg test reward:-0.028 {'win': 956, 'draw': 0, 'lose': 44}