MIT 6.S890 — Phantom Tic-Tac-Toe Challenge

Instructor: Gabriele Farina (`gfarina@mit.edu`)
TA: Zhiyuan Fan (`fanzy@mit.edu`)

Fall 2024

▸ **CONTENTS** Last update `2024-10-27`

1. Overview

Phantom Tic-Tac-Toe is a variant of traditional Tic-Tac-Toe, in which both players cannot see each other's moves. It has been a classical benchmark for testing the performance of learning algorithms. The primary objective of this project is to approximate the Nash equilibrium of the game using any suitable methods.

Despite its apparent simplicity, Phantom Tic-Tac-Toe involves approximately 27 billion distinct game states. Although the symmetric properties of the game is not taking into account, this complexity makes the precise solution of the game a bit challenging (but still possible with a good tabular implementation).

2. Game Rules

The game is played on a 3×3 grid. Players take turns, starting with Player 0, choosing a square to place their marker without knowing the current state of that square. If the square is unoccupied, the marker is placed successfully; if it is occupied, the placement fails, and the player is notified of this failure. *The turn then passes to other player.* Players are not allowed to attempt placement in a square where they have previously tried. The game progresses until one player successfully aligns three of their markers vertically, horizontally, or diagonally, thus winning the game. The squares are indexed from top left to bottom right as shown on the right.

0	1	2
3	4	5
6	7	8

3. Infoset Encoding

In Phantom Tic-Tac-Toe, a player is only aware of whether his previous moves were successful. We encode the information set in the format `| <A1> <C1> <A2> <C2> ...`, where:

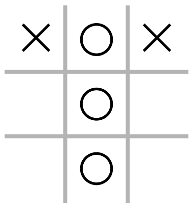
- `A1, ..., An` are the index of the squares that are attempted by the player, in order.
- `C1, ..., Cn` indicate the outcomes of the attempts:
  - `*` (asterisk) signifies a successful placement.
  - `.` (period) signifies a failed placement.

4. Sample Gameplay

Here, we present a sample gameplay. We will call information sets “infosets” for bevity:

1. Player 0 starts at the initial infoset `|` and selects square `0`, successfully placing a marker.
2. Player 1 also starts at the initial infoset `|` and selects square `4`, successfully placing a marker.
3. Player 0, now aware of the observation history `|0*`, attempts to place a marker in square `4`, which is occupied, resulting in a failed placement. Note that placing a marker in square `0` is not a valid action since they have already attempted it.
4. Player 1, with infoset `|4*`, selects square `1`, successfully placing a marker.
5. Player 0 updates the infoset to `|0*4.` and successfully places a marker in square `3`.
6. Player 1, with infoset `|4*1*`, selects square `7`, successfully placing a marker and wins by aligning `1`, `4`, and `7` vertically.

By the end of the game, the board looks like follows:



5. Automatic Evaluation of Strategies

We set up an automatic evaluation platform, which looks like the the screenshot on the right (→).

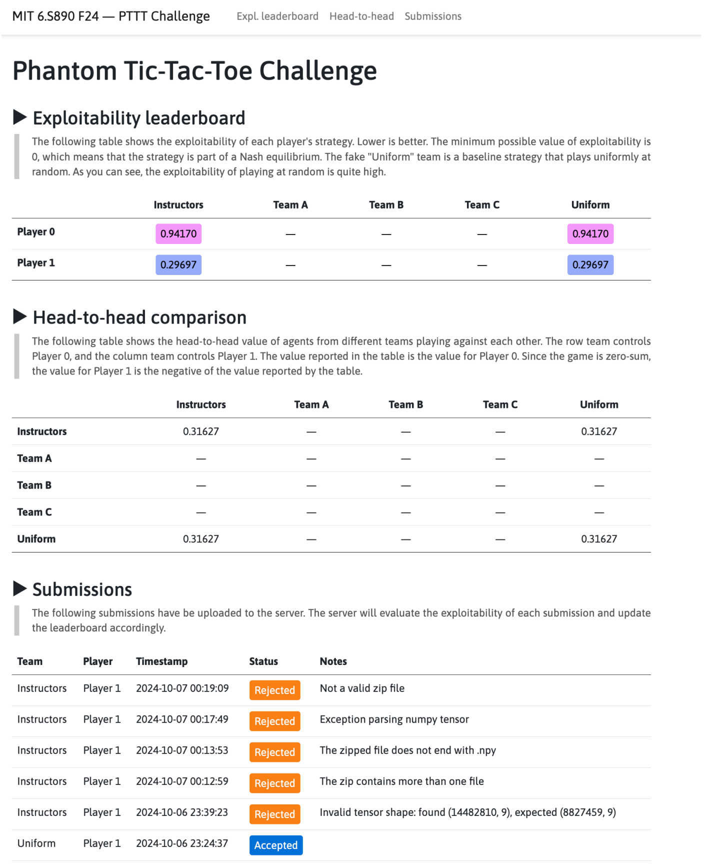
The server will first validate the strategies submitted by the teams, and then evaluate them by:

- evaluating the exploitability of the strategy, and
- computing the expected head-to-head value when playing against other teams.

The evaluation of each entry of the table takes roughly one minute, but otherwise one goal we have for this autograder is to provide feedback to the teams as quickly as possible.

Submission instructions. Each team will be handed a token they can use to upload strategies. The token will be unique to each team and should be kept confidential.

Grader link. http://6s890.lids.mit.edu:8080



Phantom Tic-Tac-Toe Challenge

▶ Exploitability leaderboard

The following table shows the exploitability of each player's strategy. Lower is better. The minimum possible value of exploitability is 0, which means that the strategy is part of a Nash equilibrium. The fake "Uniform" team is a baseline strategy that plays uniformly at random. As you can see, the exploitability of playing at random is quite high.

	Instructors	Team A	Team B	Team C	Uniform
Player 0	0.94170	—	—	—	0.94170
Player 1	0.29697	—	—	—	0.29697

▶ Head-to-head comparison

The following table shows the head-to-head value of agents from different teams playing against each other. The row team controls Player 0, and the column team controls Player 1. The value reported in the table is the value for Player 0. Since the game is zero-sum, the value for Player 1 is the negative of the value reported by the table.

	Instructors	Team A	Team B	Team C	Uniform
Instructors	0.31627	—	—	—	0.31627
Team A	—	—	—	—	—
Team B	—	—	—	—	—
Team C	—	—	—	—	—
Uniform	0.31627	—	—	—	0.31627

▶ Submissions

The following submissions have be uploaded to the server. The server will evaluate the exploitability of each submission and update the leaderboard accordingly.

Team	Player	Timestamp	Status	Notes
Instructors	Player 1	2024-10-07 00:19:09	Rejected	Not a valid zip file
Instructors	Player 1	2024-10-07 00:17:49	Rejected	Exception parsing numpy tensor
Instructors	Player 1	2024-10-07 00:13:53	Rejected	The zipped file does not end with .npy
Instructors	Player 1	2024-10-07 00:12:59	Rejected	The zip contains more than one file
Instructors	Player 1	2024-10-06 23:39:23	Rejected	Invalid tensor shape: found (14482810, 9), expected (8827459, 9)
Uniform	Player 1	2024-10-06 23:24:37	Accepted	

6. Input and Output

We ask you to submit to strategy profiles computed by your algorithm. In the provided materials, you will find two files that are respectively called `player0-infosets.txt` and `player1-infosets.txt`. The content of these files lists all possible information sets line by line for both players. Your submissions should be a zipped NumPy (`.npy`) tensor with name `pttt_pl0.zip` or `pttt_pl1.zip`, respectively. Each row of the tensor corresponds to an information set, in the order of the `player{i}-infosets.txt` files. The tensor has 9 columns, representing the probability of selecting squares `0` to `8` given each information set. Each value should be a floating-point number between 0 and 1, summing to 1 per row. **Invalid actions must have 0 probability**.

We will provide random uniform policies on the submission website.These select each valid move with equal probability at every information set.

7. Example: Construction of uniform strategies

As an example of how to construct a uniform strategy (in this case, for Player 0), consider the following Python code:

```python
import numpy as np
lines = open('player0-infoset.txt').readlines()

# Initially, set the tensor with all ones. We will mask out the illegal actions below,
# and then we will normalize row-wise just before saving the tensor below.
tensor = np.ones((len(lines), 9), dtype=np.float32)
for idx, line in enumerate(lines):
    line = line.strip() # Remove the \n
    assert line.startswith('|')
    n = len(line) - 1
    assert n % 2 == 0
    n = n // 2

    for j in range(n):
        pos = line[1 + 2 * j]
        outcome = line[2 + 2 * j]
        assert outcome in '*.'
        assert pos in '012345678'
        pos = int(pos)

        # Set zero probability to illegal actions. (Remember that we cannot probe a cell
        # more than once!)
        tensor[idx,pos] = 0

    if idx % 1000000 == 0:
        print(idx, 'done out of', len(lines))

# Renormalize row wise
tensor /= np.sum(tensor, axis=1)[:,None]
np.save('pttt_pl0.npy', tensor)
```

8. Notes

We note that achieving low exploitability may be challenging when using deep learning methods. A high level of exploitability is acceptable; however, your report should explain your methodologies and findings, including the running time of your training process. The training code should also be included in the final submission. In addition, you may find the following algorithms useful for solving the task:

- Monte Carlo CFR (MCCFR, [1]).
- Neural Fictitious Self-Play (NFSP, [2]).
- Policy-Space Response Oracle (PSRO, [3]).
- Regularized Nash Dynamics (R-NaD, [4]).
- Magnetic Mirror Descent (MMD, [5]).

Bibliography

[1] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling, "Monte Carlo sampling for regret minimization in extensive games," *Advances in neural information processing systems*, vol. 22, 2009.

[2] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *arXiv preprint arXiv:1603.01121*, 2016.

[3] M. Lanctot *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[4] J. Perolat *et al.*, "Mastering the game of Stratego with model-free multiagent reinforcement learning," *Science*, vol. 378, no. 6623, pp. 990–996, 2022.

[5] S. Sokota *et al.*, "A Unified Approach to Reinforcement Learning, Quantal Response Equilibria, and Two-Player Zero-Sum Games," in *The Eleventh International Conference on Learning Representations*,