Learning Adversarial Search algorithms

The Art of Unlosable Tic-Tac-Toe

Eric Han

June 7, 2024

Introduction

Experience

My *Singaporean* educational journey to CS, R&D:

- [2008] **Secondary School** — Informal learning; scripting, games
  - Interest in Computing: why & how computers work
- [2010] **Pioneer JC** — H2 Computing
  - Interest in research: A*STAR IHPC Quest 2009 (Bronze) - K-Means
- [2018] **B.Com. NUS** — Com. Sci w Honors
  - A*STAR Scholarship - Internships working on R&D projects
- [2024] **PhD. NUS** — AI/ML tackling scaling and robustness
  - First-author publications in AAAI, ICML.

Working experiences:

- [2018-2024] **Teaching Assistant/Graduate Tutor, NUS** — Teach UG
- [B.Com.] **Research Intern, A*STAR IHPC** — ML Platform, Rec. Sys.

Expertise

Support, teach ($>$ 500 contact hours), grade, manage/mentor tutors for:

- AI/Machine Learning
  - **CS2109s** Introduction to AI and Machine Learning
  - **CS3243** Introduction to Artificial Intelligence
- Software Engineering
  - **CS3217** Software Engineering on Modern Application Platforms
  - **CS3203** Software Engineering Project
  - **CS2030/CS2030S** Programming Methodology II

Skilled with Linux (also administration), Windows, macOS:

- **Programming Languages** — Python, C++, Java, Mojo...
- **Databases** — Firebase, SQL...
- **Typesetting / Presentation Tools** — LaTex, Markdown (This slides!)...
- **Tools/Platforms** — Git, Mlflow, Plotly, Slurm, GCP...

Effective learning is driven by an *innate desire* to learn the subject rather than *need*:

1. **Creating a relaxed and safe environment** — Informal, casual, personal.
2. **Engaging students to facilitate learning in and after class** — Telegram, buddy
3. **Creating equal opportunities for all students to learn** — Reaching out

Teaching Excellence (Tutorials/Recitation)

	2109	**2109**	**3243**	**3243**	**3217**	**3203**	**3203**	**3203**	**3203**	**3203**
Score	4.8	4.6	4.8	4.5	3.8	4.6	4.4	4.8	4.1	3.3
Resp.	36	13	25	39	6	13	16	18	20	3
Nom.	47%	30%	32%	31%	0%	61%	31%	61%	10%	33%

Plans

For the teaching position, I am interested to

- Focus on improving teaching quality
- Curriculum development/improvement
- Casual research
  - Mentor for Undergraduate Research / FYP
- Involved in consultancy/policy

Mini-Lecture

Recap on environment properties

- **Fully / Partially Observable**: Can the agent see?
- **Single / Multi-Agent**: How many agents?
- **Deterministic / Stochastic**: Is there randomness in transition?
- **Episodic / Sequential**: Is there dependence on previous action?
- **Static / Dynamic**: Can the environment change while the agent is thinking?
- **Discrete / Continuous**: Discretized or varying continuously?

Recap on formulation

Un/Informed Search (Path):

- State space
- Initial state
- Final state
- Action
- Transition

Local Search (Goal):

- Inital state
- Transition
- Heuristic/Stopping criteria

Motivation: How can we win?

Ingredients needed to formulate a problem:

- **Inital state**: Starting configuration (representation)
- **Players**: Decision-makers within the game (2 players)
- **Actions**: Potential moves that the player can make
- **Transition**: Result of a move from a state
- **Terminal/Leaf test**: Checks if the game is over
- **Utility**: Reward for a terminal state and player

Tic-tac-toe

2P childhood game where (P_O, P_X) players take turns drawing their symbols on a 3x3 grid. The winner is the first player to get 3 of his/her symbol in a row, col. or diag.

Tic-tac-toe

2P childhood game where (P_O, P_X) players take turns drawing their symbols on a 3x3 grid. The winner is the first player to get 3 of his/her symbol in a row, col. or diag.

$$
\begin{array}{c|c|c} & & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} & X & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & \\ \hline & X & \\ \hline & & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline & X & \\ \hline & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & \\ \hline X & X & \\ \hline & O & \end{array}
$$

$$
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline X & X & O \\ \hline & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline & O & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & X \end{array}
\xrightarrow{P_O} \emptyset
$$

Tic-tac-toe

2P childhood game where (P_O, P_X) players take turns drawing their symbols on a 3x3 grid. The winner is the first player to get 3 of his/her symbol in a row, col. or diag.

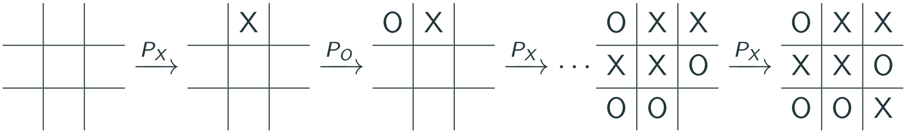
$$
\begin{array}{c|c|c} & & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} & X & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & \\ \hline & X & \\ \hline & & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline & X & \\ \hline & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & \\ \hline X & X & \\ \hline & O & \end{array}
$$

$$
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline X & X & O \\ \hline & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline & O & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & X \end{array}
\xrightarrow{P_O} \emptyset
$$

Recap — Environment Properties

Fully Observable, 2 Agent, Deterministic, Squential, Static, Discrete

Modeling Tic-tac-toe [Discussion]



2P childhood game where (P_O, P_X) players take turns drawing their symbols on a 3x3 grid. The winner is the first player to get 3 of his/her symbol in a row, col. or diag.

- **Inital state**:
- **Players**:
- **Actions**:
- **Transition**:
- **Terminal/Leaf test**:
- **Utility**:

Modeling Tic-tac-toe

$$S_0 = \begin{array}{c|c|c} 0 & 1 & 2 \\ \hline 3 & 4 & 5 \\ \hline 6 & 7 & 8 \end{array} \xrightarrow{a_0=(1,X)} \begin{array}{c|c|c} & X & \\ \hline & & \\ \hline & & \end{array} \xrightarrow{\cdots} \cdots \begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & \end{array} \xrightarrow{a_8=(8,X)} \begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & X \end{array} = S_9$$

- **Inital state**: S_0, 1D array of 9 elements: $O, X, \emptyset$
- **Players**: P_X-max, P_O-min
- **i-th Actions**: $a_i = (c_i, y) : c_i \in [0, 8]$ where $S_i[c_i] = \emptyset$, symbol $y \in X, O$
- **Transition**: $T(S_i, a_i) = S_{i+1}$, where $S_{i+1}[j] = \begin{cases} y & \text{if } j = c_i \\ S_i[j] & \text{otherwise} \end{cases}$
- **Terminal/Leaf test**: Row, col. or diag having same symbols or no moves
- **Utility**: $U(S_i, p)$ is 0 if draw, 1 if p wins, -1 if p loses

Modeling Tic-tac-toe

- **Inital state**: S_0, 1D array of 9 elements: $O, X, \emptyset$
- **Players**: P_X-max, P_O-min
- ***i*-th Actions**: $a_i = (c_i, y) : c_i \in [0, 8]$ where $S_i[c_i] = \emptyset$, symbol $y \in X, O$
- **Transition**: $T(S_i, a_i) = S_{i+1}$, where $S_{i+1}[j] = \begin{cases} y & \text{if } j = c_i \\ S_i[j] & \text{otherwise} \end{cases}$
- **Terminal/Leaf test**: $\boxed{\text{Row, col. or diag having same symbols or no moves}}$
- **Utility**: $U(S_i, p)$ is 0 if draw, 1 if p wins, -1 if p loses

FAQ: Can I describe and not write math?

Yes, but it must be **clear**; ie. Able to translate into code without additional assumptions; you should (at min) describe how the state is represented.

Modeling Tic-tac-toe in Python

- **Inital state**: S_0, 1D array of 9 elements: $O, X, \emptyset$
- ***i*-th Actions**: $a_i = (c_i, y) : c_i \in [0, 8]$ where $S_i[c_i] = \emptyset$, symbol $y \in X, O$

```
X,O,E = 'X','O','.'
SYMBOLS = {X,O}
WINNING_POS = [[0,1,2], [3,4,5], [6,7,8], [0,3,6], [1,4,7], [2,5,8], [0,4,
class TicTacToe(object):
  def __init__(self, Si=[ E ] * 9):
    self.Si = Si
    self.winner = E
  def actions(self):
    e_cis = [ ci for ci, Si_ci in enumerate(self.Si) if Si_ci == E ]
    return [ (ci, y) for y in SYMBOLS for ci in e_cis ]
```

Full code here: http://eric-han.com

Zero-sum game

Zero-sum game is a game where one player gain is equals to another's loss, where the total utility of the game is the same/constant (ie. no improvement).

Tic-tac-toe is zero-sum

- If P_X wins P_O loses: $\sum U = 1 - 1 = 0$
- If P_O wins P_X loses: $\sum U = 1 - 1 = 0$
- If P_O, P_X draws: $\sum U = 0 + 0 = 0$

So, for Tic-tac-toe: $U(S_i, X) = -U(S_i, O)$

Intuition: For nerds like me, if you played enough, you notice you keep getting draws.

Question

Can we come up with an algorithm to play Tic-tac-toe?

Tic-tac-toe gametree

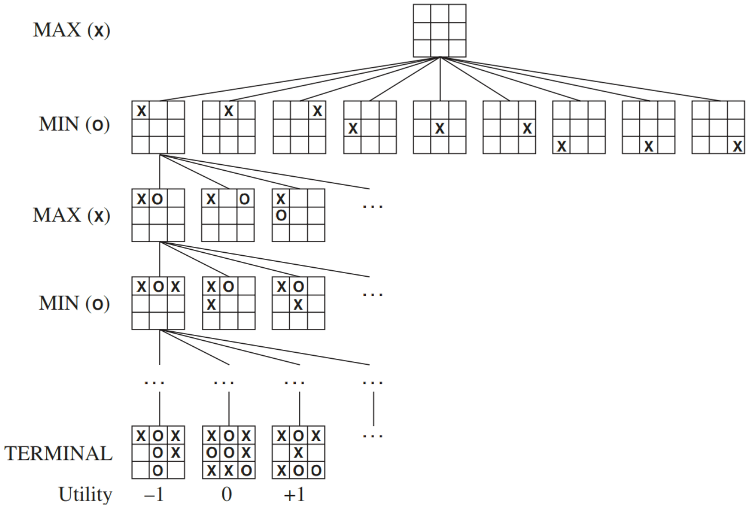


Figure 1: Gametree (R&N 3rd Ed) — Inital, Players, Actions, Transition, Terminal, Utility

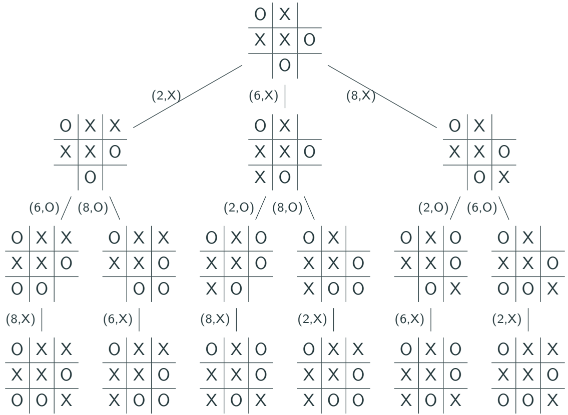
Intuition: Simulate the game until the end with an imaginary optimal opponent.

$$
\begin{array}{c|c|c} & & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} & X & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline & & \\ \hline & & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & \\ \hline & X & \\ \hline & & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & \\ \hline & X & \\ \hline & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & \\ \hline X & X & \\ \hline & O & \end{array}
$$

$$
\xrightarrow{P_O}
\boxed{
\begin{array}{c|c|c} O & X & \\ \hline X & X & O \\ \hline & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline & O & \end{array}
\xrightarrow{P_O}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & \end{array}
\xrightarrow{P_X}
\begin{array}{c|c|c} O & X & X \\ \hline X & X & O \\ \hline O & O & X \end{array}
\xrightarrow{P_O} \emptyset
}
$$

I am player P_X, trying to find the best move at $\begin{array}{c|c|c} O & X & \\ \hline X & X & O \\ \hline & O & \end{array}$

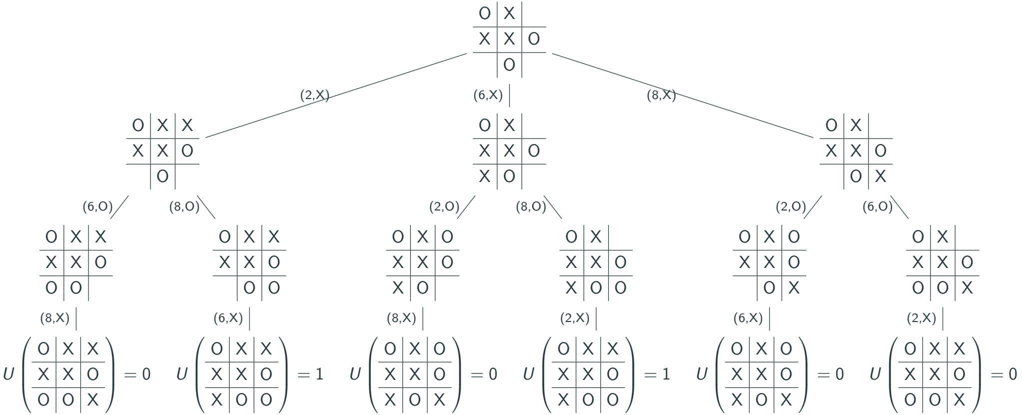
Intuition: Simulate the game until the end with an imaginary opponent.



We can fill in the utility values for the leaf nodes!

Minimax algorithm

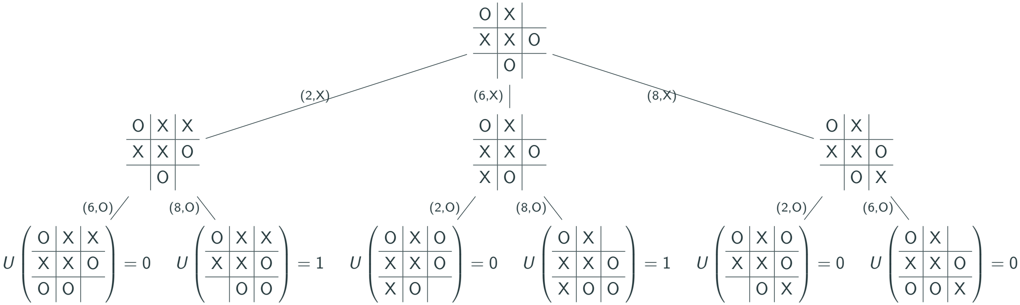
Utility for X: $U(S_i, X)$ is 0 if draw, 1 if X wins, -1 if X loses



We know we want the best action later, so we choose the best action (max) there!

Minimax algorithm

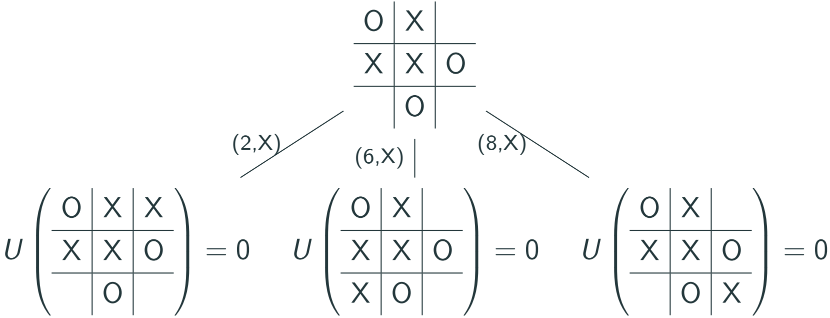
For the best action chosen, we inherit its corresponding value!



We dont know how the P_O will play this move, so

- Assume that P_O wants to win and plays optimally like me.
- We imagine that P_O chooses the best action (min) there!

Intuition: Simulate the game until the end with an imaginary *optimal* opponent.



Now I can just pick the move that is the best (max value):

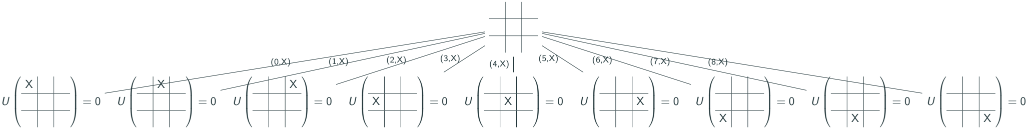
- All 3 moves would, at worse-case, end up in draws.

Minimax algorithm

Intuition: Simulate the game until the end with an imaginary *optimal* opponent.

function MINIMAX-DECISION(*state*) **returns** *an action*
return arg max $a \in$ ACTIONS(s) MIN-VALUE(RESULT(*state*, a))

Minimax Tic-tac-toe example



Minimax analysis

Issues - Huge/Infinite Game Trees

Alpha-Beta Pruning algorithm

Intuition

Alpha-Beta Tic-tac-toe example

Alpha-Beta analysis

2048

Modeling 2048

Minimax 2048 example

Minimax limitations

Expectimax algorithm

Expectimax example