Quality Diversity

Noah Syrkis

September 8, 2025

1 | Introduction

The future is a garden of forking paths [1]. Action a at state s_t yields a new state s_{t+1}. A different action a', however, might have yielded some different state s'_{t+1}.

2 | Minimax

- Suppose we have a function that:
- given a state and an action returns a new state,
- and another that given a state returns who won
- What can we do?

$$\begin{tikzcd}
 & & & a \arrow[dll] \arrow[drr] & & & & \text{min} \\
 & b \arrow[dl] \arrow[dr] & & & & c \arrow[dl] \arrow[dr] & & \text{max} \\
d & & e & & f & & g & \text{min} \\
+ & & - & & + & & + &
\end{tikzcd}$$

2 | Minimax

- Suppose we have a function that:
- given a state and an action returns a new state,
- and another that given a state returns who won
- What can we do? Play perfectly and never loose

$$\begin{tikzcd}
 & & & a \arrow[dll] \arrow[drr] & & & & \text{min} \\
 & b \arrow[dl] \arrow[dr] & & & & c \arrow[dl] \arrow[dr] & & \text{max} \\
d & & e & & f & & g & \text{min} \\
+ & & - & & + & & + &
\end{tikzcd}$$

2 | Minimax

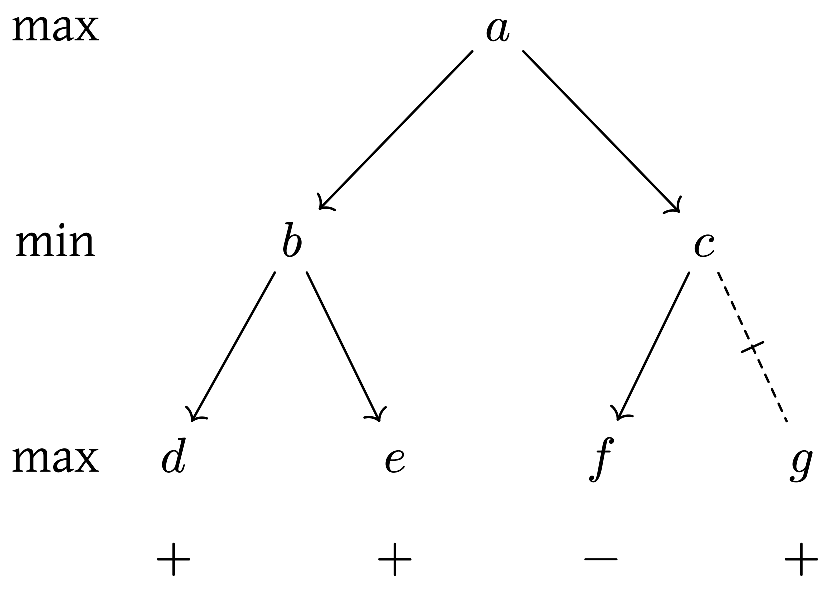
- We can win (or at least not loose) any game[1] by:
    1. Calling the minimax function for all actions
    2. Storing the values of each action in a list
    3. Taking the action with the highest value
- How can we do better? What are the issues?

Algorithm 1: minimax(node, maxim)

```
if node is terminal
    return the value of node
bestValue = $-\infty$ if maxim else $\infty$
condition = max if maxim else min
for each child of node
    value = minimax(child, not maxim)
    bestValue = condition(bestValue, value)
return bestValue
```

[1]that is two player, winnable, deterministic, etc.

3 | $\alpha - \beta$ pruning



- Skip branches worse than current floor
- α and β refer to those precisely floors

3 | $\alpha - \beta$ pruning

- Algorithm 2 looks daunting but the idea is:
- Stop exploring paths you already know are bad

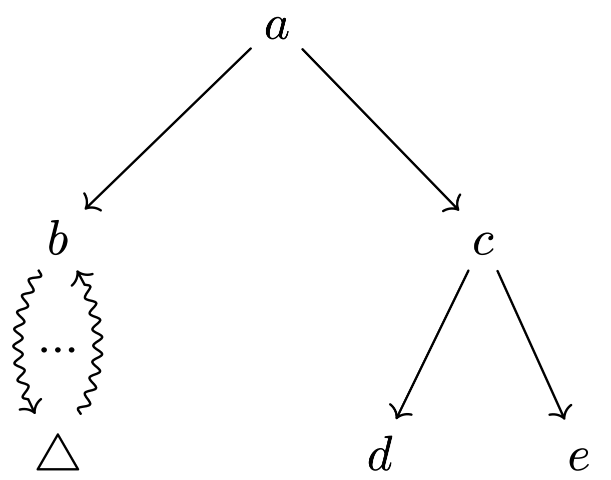
Algorithm 2: $\alpha - \beta$ pruning(node, maxim, α, β)

```
if node is terminal
    return the value of node
bestValue = $-\infty$ if maxim else $\infty$
condition = max if maxim else min
for each child of node
    value = minimax(child, not maxim, $\alpha$, $\beta$)
    bestValue = condition(bestValue, value)
    $\alpha$ = (condition($\alpha$, value) if maxim else $\alpha$)
    $\beta$ = (condition($\beta$, value) if not maxim else $\beta$)
    if $\alpha$ >= $\beta$; break
return bestValue
```

Break

4 | MCTS

- Monte Carlo (random) tree search [2]
- Core idea: sample from bottom of each branch
- How much to sample from each branch?
- How should we reach the bottom?



4.1 | Explore / exploit

- When do we exploit the best tool we have?
- When should we explore for a new tool?
- There is a good entropy based solution [3]

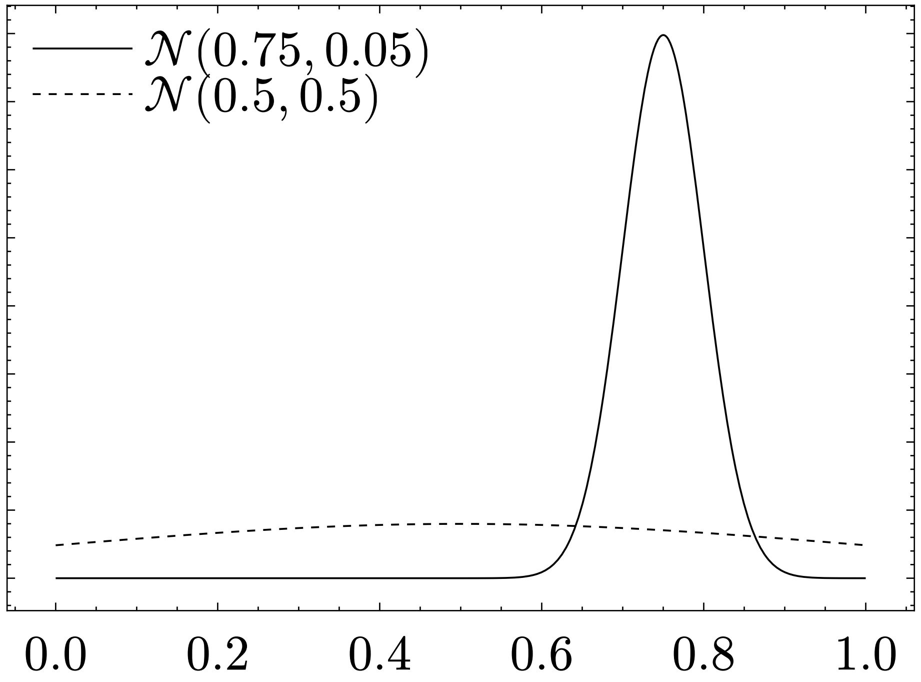


Figure 4: Which distribution would you sample from? Which is more likely to reach 1?

5 | Python

- You will see code that looks like Script 1
- In some games $s \neq o$, so we need seperate `obs`
- Multi player setup will have inner player loop

```
import gymnasium as gym
env = gym.game("tic_tac_toe")
state, done = env.init()


while not done:
  action = action_fn(state)
  state, done = env.step(state, action)
```

Script 1: Playing games in Python usually look something like this

5 | Python

- Some useful packages
- Understanding gymnasium is a must
- Get comfy with `.reset` and `.step`
- Sometimes `state` has a valid action mask!

`aigs`	package for our course

5 | Python

Dear Hajo

5 | Python

Sebastian can't fund my trip since our money is from the Swiss project can't use for this. So, this email is about me convening you that ALife 2025 is a relevant conference:

Regarding general relevance: I believe everyone or almost everyone in my lab will be attending, so it is very much within our wheelhouse.

Relevance for me specifically: I'll be presenting a poster on the use of Uiua (www.uiua.org)–a stack-based array language written in Rust–for artificial life research and creative computing in general. The submission on which I've been accepted is attached here for reference.

I have shown examples of its potential use in graphical art on my website (virian.com/rhos). Uiua allows for much more expressive and consice code, making protoyping fast, and visual exceptionally quick.

Two anecdotes on the relevance of Uiua in particular:

5 | Python

1) I've recently had a conversation with Kathrine Lotz, institute leader of the Royal Architecture School here, and she was very intersted in in the context of creative computing.
2) As my github shows my Uiua projects I've been contacted by a headhunter for one of the big AI firms to do weekly coding sessions to help them have their models learn Uiua better. Given Uiue's expressiveness, the same code can be written with very few tokens by LLMs compared to a traditional language like python.
What are your thoughts on this?
– Noah Syrkis

References

[1] J. L. Borges, "The Garden of Forking Paths," Ficciones. Grove Press, New York, 1962.

[2] C. B. Browne et al., "A Survey of Monte Carlo Tree Search Methods," IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1–43, Mar. 2012, doi: 10.1109/TCIAIG.2012.2186810.

[3] H. Robbins, "SOME ASPECTS OF THE SEQUENTIAL DESIGN OF EXPERIMENTS," 1952.