Games III: Alphabeta

15-150 M21

Lecture 0728-0
28 July 2021

0 Optimizing Minimax

The MiniMax Algorithm

Fix a search depth d.

- `evaluate`:
  - If at depth d, call the estimator to obtain the value of the current state
  - Otherwise, `search` through the available moves to find the best move (for the current player). The value of the current state is the value of the state resulting from that move.
- `search`: Make a sequence containing all the game states that would result from making a single (legal) move from the current state, `evaluate` all them, and pick the best one (for the current player)

Demonstration:

Minimax

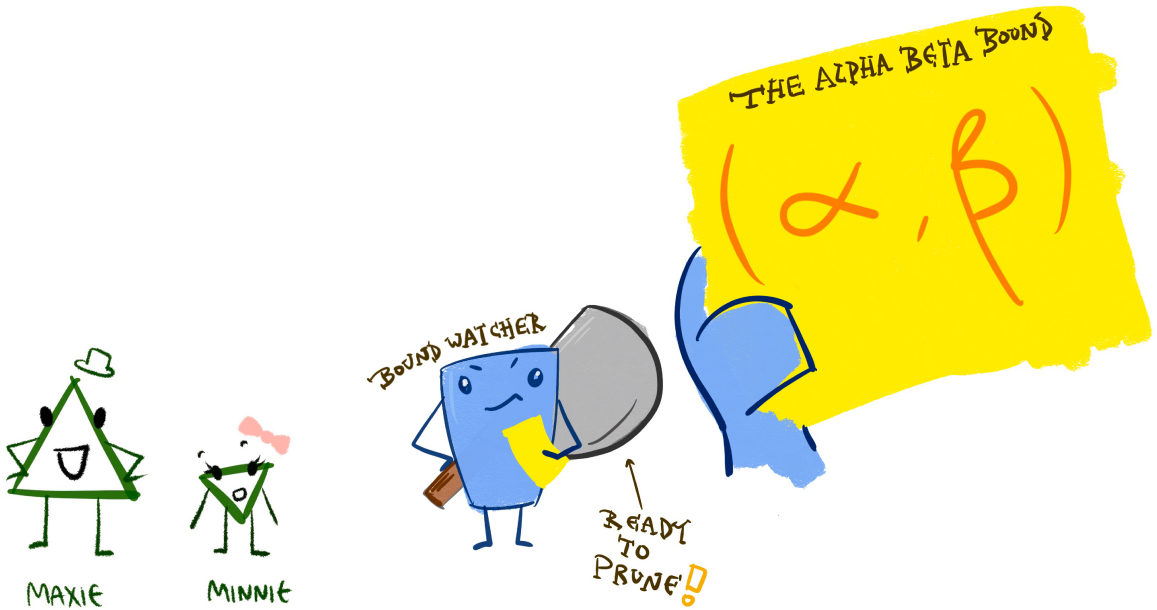
Advantages of Minimax:

- Correctly determines optimal play
- Massively parallelizable

Disadvantages of Minimax:

- Huge amount of work
- Indeed, often performs unnecessary computation

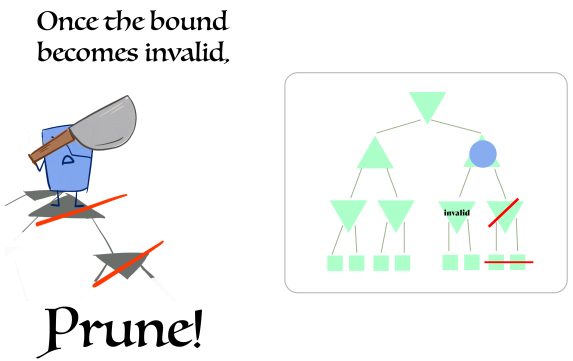
As we perform MiniMax, we want to keep track of "what can be guaranteed" to inform us when we're exploring an irrelevant subtree.



Idea

So, for every point along the minimax algorithm, there is some estimator guess value α, which represents the greatest value that **Maxie** can *guarantee*. Analogously, we'll keep track of some value β representing the least value that **Minnie** can guarantee. It must be the case that $\alpha \leq \beta$.

When **Minnie** encounters a node whose value is $\leq \alpha$, then she can "prune" the rest of the current subtree: **Maxie** won't let the game get to this point. If **Maxie** encounters a node whose value is $\geq \beta$, then she prunes.



Fix a search depth d.

- `evaluate`:
  - If at depth d, call the estimator to obtain the value of the current state
  - Otherwise, `search` through the available moves to find the best move (for the current player). The value of the current state is the value of the state resulting from that move.
- `search`: Try possible moves one-by-one,
  - `evaluate` the resulting game state (passing down the current (α, β)) to obtain x
  - Update α or update β or prune (as called for in the chart)
  - If we evaluate all the moves without pruning, then pick the best one (for the current player)

	$x \leq \alpha$	$\alpha < x < \beta$	$\beta \leq x$
Maxie	Ignore	Update α	Prune
Minnie	Prune	Update β	Ignore

Demonstration:

Alphabeta

Helpful: Seq.showl

In the SEQ signature:

```
datatype 'a lview =
    Nil | Cons of 'a * 'a seq
```

```
val showl : 'a seq -> 'a lview
val hidel : 'a lview -> 'a seq
```

5 minute break