Inf2D 04: Adversarial Search

Michael Herrmann

University of Edinburgh, School of Informatics

24/01/2017

Credits: The content of this lecture was prepared by Michael Rovatsos and follows R&N
Outline

- Games
- Optimal decisions
- $\alpha - \beta$ pruning
- Imperfect, real-time decisions
We are (usually) interested in zero-sum games of perfect information

- Deterministic, fully observable
- Agents act alternately
- Utilities at end of game are equal and opposite

“Unpredictable” opponent ➔ specifying a move for every possible opponent reply

Time limits ➔ unlikely to find goal, must approximate
Game tree (2-player, deterministic, turns)

- 2 players: MAX and MIN
- MAX moves first
- Tree built from MAX’s POV

← Utility of each terminal state from MAX’s point of view.
Normal search: optimal decision is a sequence of actions leading to a goal state (i.e. a winning terminal state)

Adversarial search:
- MIN has a say in game
- MAX needs to find a contingent strategy which specifies:
  - MAX’s move in initial state then ...
  - MAX’s moves in states resulting from every response by MIN to the move then ...
  - MAX’s moves in states resulting from every response by MIN to all those moves, etc. ...

\[
\text{minimax value of a node=utility for MAX of being in corresponding state:} \\
\text{\( MINIMAX(s) = \)} \\
\begin{cases} 
\text{UTILITY}(s) & \text{if } \text{TERMINAL-TEST}(s) \\
\max_{a \in \text{Actions}(s)} \text{MINIMAX} \left( \text{RESULT} \left( s, a \right) \right) & \text{if } \text{PLAYER}(s) = \text{MAX} \\
\min_{a \in \text{Actions}(s)} \text{MINIMAX} \left( \text{RESULT} \left( s, a \right) \right) & \text{if } \text{PLAYER}(s) = \text{MIN}
\end{cases}
\]
Perfect play for deterministic games

Idea: choose move to position with highest minimax value = best achievable payoff against best play

Example: 2-ply game:
Minimax algorithm

**function** MINIMAX-DECISION(state) **returns** an action

\[
\text{return } \arg \max_{a \in \text{Actions}(s)} \text{MIN-VALUE(RESULT(state, a))}
\]

**function** MAX-VALUE(state) **returns** a utility value

\[
\text{if TERMINAL-TEST(state) then return UTILITY(state)} \\
\quad v \leftarrow -\infty \\
\quad \text{for each } a \text{ in ACTIONS(state) do} \\
\quad \quad v \leftarrow \max(v, \text{MIN-VALUE(RESULT(s, a))}) \\
\text{return } v
\]

**function** MIN-VALUE(state) **returns** a utility value

\[
\text{if TERMINAL-TEST(state) then return UTILITY(state)} \\
\quad v \leftarrow \infty \\
\quad \text{for each } a \text{ in ACTIONS(state) do} \\
\quad \quad v \leftarrow \min(v, \text{MAX-VALUE(RESULT(s, a))}) \\
\text{return } v
\]

Idea: Proceed all the way down to the leaves of the tree then minimax values are backed up through tree
Properties of minimax

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes (against an optimal opponent)
- **Time complexity?** $O(b^m)$
- **Space complexity?** $O(bm)$ (depth-first exploration)

For chess, $b \approx 35$, $m \approx 100$ for “reasonable” games
  - exact solution completely infeasible!
  - would like to eliminate (large) parts of game tree
\( \alpha - \beta \) pruning example
$\alpha$-$\beta$ pruning example

```
MAX

MIN

3
12
8
2
X
X

\geq 3
\leq 2
```
α-β pruning example
$\alpha - \beta$ pruning example

```
MAX

MIN

3  12  8  2  14  5
```

```
3

3

X

X

≥ 3

≤ 2

≤ 5
```
\( \alpha - \beta \) pruning example
Are minimax value of root and, hence, minimax decision independent of pruned leaves?

Let pruned leaves have values $u$ and $v$, then

\[
\text{MINIMAX}(\text{root}) = \max(\min(3, 12, 8), \min(2, u, v), \min(14, 5, 2))
\]
\[
= \max(3, \min(2, u, v), 2)
\]
\[
= \max(3, z, 2) \text{ where } z \leq 2
\]
\[
= 3
\]

Yes!
Pruning does not affect final result (as we saw for example)

**Good move ordering** improves effectiveness of pruning (How could previous tree be better?)

With “perfect ordering”, time complexity $O\left(b^{m/2}\right)$
- branching factor goes from $b$ to $\sqrt{b}$
- (alternative view) doubles depth of search compared to minimax

A simple example of the value of reasoning about which computations are relevant (a form of **meta-reasoning**)
Why is it called $\alpha$-$\beta$?

- $\alpha$ is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for MAX.
- If $v$ is worse than $\alpha$, MAX will avoid it → prune that branch.
- Define $\beta$ similarly for MIN.
The $\alpha$-$\beta$ algorithm

```
function ALPHA-BETA-SEARCH(state) returns an action
    $v \leftarrow \text{MAX-VALUE}(state, -\infty, +\infty)$
    return the action in ACTIONS(state) with value $v$

function MAX-VALUE(state, $\alpha$, $\beta$) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow -\infty$
    for each $a$ in ACTIONS(state) do
        $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
        if $v \geq \beta$ then return $v$
        $\alpha \leftarrow \text{MAX}(\alpha, v)$
    return $v$
```

- $\alpha$ is value of the best i.e. highest-value choice found so far at any choice point along the path for MAX
- $\beta$ is value of the best i.e. lowest-value choice found so far at any choice point along the path for MIN
The $\alpha$-$\beta$ algorithm

```
function MIN-VALUE(state, $\alpha$, $\beta$) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow +\infty$
    for each $a$ in ACTIONS(state) do
        $v \leftarrow \min(v, \text{MAX-VALUE(RESULT}(s, a), \alpha, \beta))$
        if $v \leq \alpha$ then return $v$
        $\beta \leftarrow \min(\beta, v)$
    return $v$
```

Prune as this value is worse for MAX and so won't ever be chosen by MAX!
Suppose we have 100 secs, explore $10^4$ nodes/sec
   $\Rightarrow 10^6$ nodes per move

Standard approach:
   - cutoff test: e.g., depth limit (perhaps add quiescence search, which tries to search interesting positions to a greater depth than quiet ones)

   evaluation function
   $= \text{estimated desirability of position}$
Evaluation functions

- For chess, typically linear weighted sum of features

\[ EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) \]

where each \( w_i \) is a weight and each \( f_i \) is a feature of state \( s \)

- Example
  - queen = 1, king = 2, etc.
  - \( f_i \): number of pieces of type \( i \) on board
  - \( w_i \): value of the piece of type \( i \)
Minimax Cutoff is identical to MinimaxValue except

1. TERANL-TEST is replaced by CUTOFF
2. UTILITY is replaced by EVAL

Does it work in practice?

\[ b^m = 10^6, \ b = 35 \implies m = 4 \]

4-ply lookahead is a hopeless chess player!

- 4-ply \( \approx \) human novice
- 8-ply \( \approx \) typical PC, human master
- 12-ply \( \approx \) Deep Blue, Kasparov
Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.


Othello: human champions refuse to compete against computers, who are too good.

Go: human champions used to refuse to compete against computers, who are too bad. In Go, \( b > 300 \), so most programs use pattern knowledge bases to suggest plausible moves. 2016: AlphaGo
Games are fun to work on!
They illustrate several important points about AI
Perfection is unattainable → must approximate
Good idea to think about what to think about