CSE 473: Introduction to Artificial Intelligence

Hanna Hajishirzi
Adversarial Search

slides adapted from
Dan Klein, Pieter Abbeel ai.berkeley.edu
And Dan Weld, Luke Zettelmoyer
Announcements

- Written HW1 is released: (due: 10/23)
  - Start ASAP.
- Project 2 is released: (due 10/30)
  - About games: Start ASAP.
Adversarial Search
Value of a State

Value of a state: The best achievable outcome (utility) from that state

Non-Terminal States:
$$V(s) = \max_{s' \in \text{children}(s)} V(s')$$

Terminal States:
$$V(s) = \text{known}$$
Minimax Values

**States Under Agent’s Control:**

\[ V(s) = \max_{s' \in \text{successors}(s)} V(s') \]

**States Under Opponent’s Control:**

\[ V(s') = \min_{s \in \text{successors}(s')} V(s) \]

**Terminal States:**

\[ V(s) = \text{known} \]
Minimax Implementation (Dispatch)

```python
def value(state):
    if the state is a terminal state: return the state’s utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)

def max-value(state):
    initialize v = -\infty
    for each successor of state:
        v = max(v, value(successor))
    return v

def min-value(state):
    initialize v = +\infty
    for each successor of state:
        v = min(v, value(successor))
    return v
```
Minimax Example
Minimax Properties

Optimal against a perfect player. Otherwise?
Video of Demo Min vs. Exp (Min)
Video of Demo Min vs. Exp (Exp)
Minimax Efficiency

- How efficient is minimax?
  - Just like (exhaustive) DFS
  - Time: $O(b^m)$
  - Space: $O(bm)$

- Example: For chess, $b \approx 35$, $m \approx 100$
  - Exact solution is completely infeasible
  - But, do we need to explore the whole tree?
Resource Limits
Game Tree Pruning
Minimax Example
**Alpha-Beta Pruning**

- **General configuration (MIN version)**
  - We’re computing the MIN-VALUE at some node $n$
  - We’re looping over $n$’s children
  - $n$’s estimate of the childrens’ min is dropping
  - Who cares about $n$’s value? MAX
  - Let $a$ be the best value that MAX can get at any choice point along the current path from the root
  - If $n$ becomes worse than $a$, MAX will avoid it, so we can stop considering $n$’s other children (it’s already bad enough that it won’t be played)

- **MAX version is symmetric**
def min-value(state, α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
    if v ≤ β return v
    β = min(β, v)
    return v

def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
    if v ≥ β return v
    α = max(α, v)
    return v
Alpha-Beta Pruning Properties

- This pruning has **no effect** on minimax value computed for the root!

- Values of intermediate nodes might be wrong
  - Important: children of the root may have the wrong value
  - So the most naïve version won’t let you do action selection

- Good child ordering improves effectiveness of pruning

- With “perfect ordering”:
  - Time complexity drops to $O(b^{m/2})$
  - Doubles solvable depth!
  - Full search of, e.g. chess, is still hopeless…

- This is a simple example of **metareasoning** (computing about what to compute)
Alpha-Beta Quiz
Alpha-Beta Quiz 2

Diagram:

- Nodes labeled with values:
  - a: 10
  - b: 2
  - c: 10
  - d: 6
  - e: 100
  - f: 8
  - g: 1
  - h: 2
  - i: 2
  - j: 20
  - k: 4

- At node a, the value is 10, which is greater than or equal to 100. Therefore, the path is pruned.
- At node b, the value is 2, which is less than or equal to 2. Therefore, the path is pruned.
- At node e, the value is 100, which is greater than or equal to 100. Therefore, the path is pruned.
- At node g, the value is 1, which is less than 2. Therefore, the path is pruned.

No further paths are pruned as the values meet the conditions specified.
Recap: Minimax
Resource Limits – Game Prunning
Alpha-Beta Pruning

- General configuration (MIN version)
  - We’re computing the MIN-VALUE at some node $n$
  - We’re looping over $n$’s children
  - $n$’s estimate of the childrens’ min is dropping
  - Who cares about $n$’s value? MAX
  - Let $a$ be the best value that MAX can get at any choice point along the current path from the root
  - If $n$ becomes worse than $a$, MAX will avoid it, so we can stop considering $n$’s other children (it’s already bad enough that it won’t be played)

- MAX version is symmetric
Alpha-Beta Pruning Properties

- This pruning has **no effect** on minimax value computed for the root!

- Values of intermediate nodes might be wrong
  - Important: children of the root may have the wrong value
  - So the most naive version won’t let you do action selection

- Good child ordering improves effectiveness of pruning
Alpha-Beta Quiz
Resource Limits
Resource Limits

- Problem: In realistic games, cannot search to leaves!

- Solution: Depth-limited search
  - Instead, search only to a limited depth in the tree
  - Replace terminal utilities with an evaluation function for non-terminal positions

- Example:
  - Suppose we have 100 seconds, can explore 10K nodes/sec
  - So can check 1M nodes per move
  - α-β reaches about depth 8 – decent chess program

- Guarantee of optimal play is gone

- More plies makes a BIG difference

- Use iterative deepening for an anytime algorithm
Depth Matters

- Evaluation functions are always imperfect
- The deeper in the tree the evaluation function is buried, the less the quality of the evaluation function matters
- An important example of the tradeoff between complexity of features and complexity of computation

[Demo: depth limited (L6D4, L6D5)]
Video of Demo Limited Depth (2)
Video of Demo Limited Depth (10)
Evaluation Functions
Evaluation Functions

- Evaluation functions score non-terminals in depth-limited search

- Ideal function: returns the actual minimax value of the position
- In practice: typically weighted linear sum of features:

\[ \text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) \]

- e.g. \( f_1(s) = (\text{num white queens} - \text{num black queens}) \), etc.
Evaluation for Pacman
Video of Demo Thrashing (d=2)
A danger of replanning agents!
- He knows his score will go up by eating the dot now (west, east)
- He knows his score will go up just as much by eating the dot later (east, west)
- There are no point-scoring opportunities after eating the dot (within the horizon, two here)
- Therefore, waiting seems just as good as eating: he may go east, then back west in the next round of replanning!
Video of Demo Thrashing -- Fixed (d=2)
Video of Smart Ghosts (Coordination)
Video of Demo Smart Ghosts (Coordination) – Zoomed In
Synergies between Alpha-Beta and Evaluation Function

- Alpha-Beta: amount of pruning depends on expansion ordering
  - Evaluation function can provide guidance to expand most promising nodes first

- Alpha-beta:
  - Value at a min-node will only keep going down
  - Once value of min-node lower than better option for max along path to root, can prune
  - Hence, IF evaluation function provides upper-bound on value at min-node, and upper-bound already lower than better option for max along path to root THEN can prune