

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Adversarial Search

Many slides courtesy of Dan Klein, Stuart Russell, or Andrew Moore

CS 5300 / CS 6300
Artificial Intelligence
Spring 2009

Hal Daumé III
hal@cs.utah.edu

www.cs.utah.edu/~hal/courses/2009S_AI

Slide 1

CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Announcements

- Homework 2 has been posted since Tue
 - Due next week
- So has HW1 solution
 - Grades hopefully done early next week
- Project 2 (Pacman with ghosts) up soon
 - Best case tomorrow, worst case Monday
 - Due 17 Feb (about 2.5 weeks)
- Office hours on the web page
 - Me: Tue (10-11) and Thr (4:30-5:30)
 - Scott: Mon (3-4), Tue (2-3:30) and Wed (12-1:30)
- Syllabus slightly tweaked (but due dates unaffected)

Slide 2

CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Adversarial Search



[DEMO: mystery pacman]

Slide 3

CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Game Playing State-of-the-Art

- Checkers:** Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions. Checkers is now solved!
- Chess:** Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue examined 200 million positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply.
- Othello:** human champions refuse to compete against computers, which are too good.
- Go:** human champions refuse to compete against computers, which are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.
- Pacman:** unknown

Slide 4

CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

GamesCrafters



http://gamescrafters.berkeley.edu/

Slide 5

CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Game Play Examples?

- Many differ
 - Deterministic, 1 player, perfect information?
 - Deterministic, 1 player, imperfect information?
- Axes:
 - Deterministic, >1 player, perfect information?
 - Deterministic, >1 player, imperfect information?
- One, two o
 - Stochastic, 1 player, perfect information?
 - Stochastic, 1 player, imperfect information?
- Perfect info
 - Stochastic, >1 player, perfect information?
 - Stochastic, >1 player, imperfect information?
- Want algorithm which recommends a move in each state

Slide 6

CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Deterministic Games

- Many possible formalizations, one is:
 - States: S (start at s_0)
 - Players: $P=\{1\dots N\}$ (usually take turns)
 - Actions: A (may depend on player / state)
 - Transition Function: $S \times A \rightarrow S$
 - Terminal Test: $S \rightarrow \{t, f\}$
 - Terminal Utilities: $S \times P \rightarrow R$
- Solution for a player is a policy: $S \rightarrow A$

Slide 7 CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Deterministic Single-Player?

- Deterministic, single player, perfect information:
 - Know the rules
 - Know what actions do
 - Know when you win
 - E.g. Freecell, 8-Puzzle, Rubik's cube
 - ... it's just search!
- Slight reinterpretation:
 - Each node stores a **value**: the best outcome it can reach
 - This is the maximal outcome of its children
 - Note that we don't have path sums as before (utilities at end)
 - After search, can pick move that leads to best node

Slide 8 CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Deterministic Two-Player

- E.g. tic-tac-toe, chess, checkers
- **Minimax search**
 - A state-space search tree
 - Players alternate
 - Each layer, or ply, consists of a round of moves
 - Choose move to position with highest **minimax value** = best achievable utility against best play
- Zero-sum games
 - One player maximizes result
 - The other minimizes result

Slide 9 CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Tic-tac-toe Game Tree

Slide 10 Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Minimax Example

Slide 11 CS 5300: Adversarial Search

UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Minimax Search

function **MAX-VALUE**(state) returns a utility value
 if **TERMINAL-TEST**(state) then return **UTILITY**(state)
 $v \leftarrow -\infty$
 for a, s in **SUCCESSORS**(state) do $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$
 return v

function **MIN-VALUE**(state) returns a utility value
 if **TERMINAL-TEST**(state) then return **UTILITY**(state)
 $v \leftarrow \infty$
 for a, s in **SUCCESSORS**(state) do $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$
 return v

Slide 12 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Minimax Properties

- Optimal against a perfect player. Otherwise?
- Time complexity?
 - $O(b^m)$
- Space complexity?
 - $O(bm)$
- For chess, $b \approx 35$, $m \approx 100$
 - Exact solution is completely infeasible
 - But, do we need to explore the whole tree?

Slide 13 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Resource Limits

- Cannot search to leaves
- Depth-limited search
 - Instead, search a limited depth of tree
 - Replace terminal utilities with an eval function for non-terminal positions
- Guarantee of optimal play is gone
- More plies makes a BIG difference
 - [DEMO: limitedDepth]
- 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

Slide 14 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Evaluation Functions

- Function which scores non-terminals

- Ideal function: returns the utility of the position
- In practice: typically weighted linear sum of features:
- e.g. $f_i(s) = (\text{num white queens} - \text{num black queens}), \text{ etc.}$

Slide 15 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Iterative Deepening

Iterative deepening uses DFS as a subroutine:

- Do a DFS which only searches for paths of length 1 or less. (DFS gives up on any path of length 2)
- If "1" failed, do a DFS which only searches paths of length 2 or less.
- If "2" failed, do a DFS which only searches paths of length 3 or less.and so on.

This works for single-agent search as well!

Why do we want to do this for multiplayer games?

Slide 17 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Pruning in Minimax Search

Slide 18 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

α - β Pruning

- General configuration
 - α is the best value that MAX can get at any choice point along the current path
 - If n becomes worse than α , MAX will avoid it, so can stop considering n 's other children
 - Define β similarly for MIN

Slide 20 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

α - β Pruning Pseudocode

```

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow -\infty$ 
  for  $a, s$  in SUCCESSORS(state) do  $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$ 
  return  $v$ 

function MAX-VALUE(state,  $\alpha, \beta$ ) returns a utility value
  inputs: state, current state in game
          $\alpha$ , the value of the best alternative for MAX along the path to state
          $\beta$ , the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow -\infty$ 
  for  $a, s$  in SUCCESSORS(state) do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta))$ 
    if  $v \geq \beta$  then return  $v$ 
     $\alpha \leftarrow \text{MAX}(\alpha, v)$ 
  return  $v$ 

```

Slide 21 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

α - β Pruning Properties

- Pruning has **no effect** on final result
- Good move 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!
- A simple example of **metareasoning**, here reasoning about which computations are relevant

Slide 22 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Non-Zero-Sum Games

- Similar to minimax:
- Utilities are now tuples
- Each player maximizes their own entry at each node
- Propagate (or back up) nodes from children

Slide 23 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Stochastic Single-Player

- What if we don't know what the result of an action will be? E.g.,
 - In solitaire, shuffle is unknown
 - In minesweeper, mine locations
 - In pacman, ghosts!
- Can do **expectimax search**
 - Chance nodes, like actions except the environment controls the action chosen
 - Calculate utility for each node
 - Max nodes as in search
 - Chance nodes take average (expectation) of value of children
- Later, we'll learn how to formalize this as a **Markov Decision Process**

Slide 24 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Stochastic Two-Player

- E.g. backgammon
- Expectiminimax (!)
- Environment is an extra player that moves after each agent
- Chance nodes take expectations, otherwise like minimax

```

if state is a MAX node then
  return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a MIN node then
  return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a chance node then
  return average of EXPECTIMINIMAX-VALUE of SUCCESSORS(state)

```

Slide 25 CS 5300: Adversarial Search

UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Stochastic Two-Player

- Dice rolls increase b : 21 possible rolls with 2 dice
- Backgammon ≈ 20 legal moves
- Depth 4 = $20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$
- As depth increases, probability of reaching a given node shrinks
 - So value of lookahead is diminished
 - So limiting depth is less damaging
 - But pruning is less possible...
- TDGammon uses depth-2 search + very good eval function + reinforcement learning: world-champion level play

Slide 26 CS 5300: Adversarial Search

What's Next?

- Make sure you know what:
 - Probabilities are
 - Expectations are
 - You should have no trouble answering any question from:
 - www.eng.utah.edu/~cs2100/InClassExercises/ice{13,14,15}.pdf
 - If you do, review week 10/11 lectures:
 - www.eng.utah.edu/~cs2100/schedule.html
- Next topics:
 - Dealing with uncertainty
 - How to learn evaluation functions
 - Markov Decision Processes