CS 188: Artificial Intelligence

Search with other Agents II

Instructor: Anca Dragan, Sergey Levine

University of California, Berkeley

[These slides adapted from Dan Klein and Pieter Abbeel]
Minimax Example
Minimax Example
Minimax Example

Diagram of a minimax tree with values at each node.
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
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.
Other Game Types
Multi-Agent Utilities

- What if the game is not zero-sum, or has multiple players?

- Generalization of minimax:
  - Terminals have utility tuples
  - Node values are also utility tuples
  - Each player maximizes its own component
  - Can give rise to cooperation and competition dynamically…
Uncertain Outcomes
Worst-Case vs. Average Case

Idea: Uncertain outcomes controlled by chance, not an adversary!
Expectimax Search

- Why wouldn’t we know what the result of an action will be?
  - Explicit randomness: rolling dice
  - Unpredictable opponents: the ghosts respond randomly
  - Unpredictable humans: humans are not perfect
  - Actions can fail: when moving a robot, wheels might slip

- Values should now reflect average-case (expectimax) outcomes, not worst-case (minimax) outcomes

- **Expectimax search**: compute the average score under optimal play
  - Max nodes as in minimax search
  - Chance nodes are like min nodes but the outcome is uncertain
  - Calculate their expected utilities
  - I.e. take weighted average (expectation) of children

- Later, we’ll learn how to formalize the underlying uncertain-result problems as Markov Decision Processes

[Demo: min vs exp (L7D1,2)]
Video of Demo Minimax vs Expectimax (Min)
Video of Demo Minimax vs Expectimax (Exp)
Expectimax Pseudocode

**def value(state):**
- if the state is a terminal state: return the state’s utility
- if the next agent is **MAX**: return \( \max - \text{value}(state) \)
- if the next agent is **EXP**: return \( \exp - \text{value}(state) \)

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

**def exp-value(state):**
- initialize \( v = 0 \)
- for each successor of state:
  - \( p = \text{probability}(successor) \)
  - \( v += p \times \text{value}(successor) \)
- return \( v \)
def exp_value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v

v = (1/2) (8) + (1/3) (24) + (1/6) (-12) = 10
Expectimax Example

Diagram of a decision tree with numerical values at the nodes.
Expectimax Pruning?
Depth-Limited Expectimax

Estimate of true expectimax value (which would require a lot of work to compute)
Probabilities
A random variable represents an event whose outcome is unknown.

A probability distribution is an assignment of weights to outcomes.

Example: Traffic on freeway
- Random variable: T = whether there’s traffic
- Outcomes: T in {none, light, heavy}
- Distribution: P(T=none) = 0.25, P(T=light) = 0.50, P(T=heavy) = 0.25

Some laws of probability (more later):
- Probabilities are always non-negative
- Probabilities over all possible outcomes sum to one

As we get more evidence, probabilities may change:
- P(T=heavy) = 0.25, P(T=heavy | Hour=8am) = 0.60
- We’ll talk about methods for reasoning and updating probabilities later
**Reminder: Expectations**

- The expected value of a function of a random variable is the average, weighted by the probability distribution over outcomes.

- Example: How long to get to the airport?

<table>
<thead>
<tr>
<th>Time</th>
<th>Probability</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 min</td>
<td>0.25</td>
<td>$0.25 \times 20$ min</td>
</tr>
<tr>
<td>30 min</td>
<td>0.50</td>
<td>$0.50 \times 30$ min</td>
</tr>
<tr>
<td>60 min</td>
<td>0.25</td>
<td>$0.25 \times 60$ min</td>
</tr>
</tbody>
</table>

Total expected time: $35$ min
What Probabilities to Use?

- In expectimax search, we have a probabilistic model of how the opponent (or environment) will behave in any state:
  - Model could be a simple uniform distribution (roll a die)
  - Model could be sophisticated and require a great deal of computation
  - We have a chance node for any outcome out of our control: opponent or environment
  - The model might say that adversarial actions are likely!

- For now, assume each chance node magically comes along with probabilities that specify the distribution over its outcomes.

Having a probabilistic belief about another agent’s action does not mean that the agent is flipping any coins!
Quiz: Informed Probabilities

- Let’s say you know that your opponent is actually running a depth 2 minimax, using the result 80% of the time, and moving randomly otherwise.
- Question: What tree search should you use?

**Answer: Expectimax!**

- To figure out EACH chance node’s probabilities, you have to run a simulation of your opponent.
- This kind of thing gets very slow very quickly.
- Even worse if you have to simulate your opponent simulating you…
- … except for minimax and maximax, which have the nice property that it all collapses into one game tree.

This is basically how you would model a human, except for their utility: their utility might be the same as yours (i.e. you try to help them, but they are depth 2 and noisy), or they might have a slightly different utility (like another person navigating in the office).
Modeling Assumptions
The Dangers of Optimism and Pessimism

**Dangerous Optimism**
Assuming chance when the world is adversarial

**Dangerous Pessimism**
Assuming the worst case when it’s not likely
Pacman used depth 4 search with an eval function that avoids trouble.
Ghost used depth 2 search with an eval function that seeks Pacman.

Results from playing 5 games

<table>
<thead>
<tr>
<th></th>
<th>Adversarial Ghost</th>
<th>Random Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimax Pacman</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expectimax Pacman</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Demos: world assumptions (L7D3,4,5,6)]
Assumptions vs. Reality

Pacman used depth 4 search with an eval function that avoids trouble
Ghost used depth 2 search with an eval function that seeks Pacman

<table>
<thead>
<tr>
<th></th>
<th>Adversarial Ghost</th>
<th>Random Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimax</strong></td>
<td>Won 5/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Pacman</td>
<td>Avg. Score: 483</td>
<td>Avg. Score: 493</td>
</tr>
<tr>
<td><strong>Expectimax</strong></td>
<td>Won 1/5</td>
<td>Won 5/5</td>
</tr>
</tbody>
</table>

Results from playing 5 games

[Demos: world assumptions (L7D3,4,5,6)]
Video of Demo World Assumptions
Random Ghost – Expectimax Pacman
Video of Demo World Assumptions
Adversarial Ghost – Minimax Pacman
Video of Demo World Assumptions
Adversarial Ghost – Expectimax Pacman
Video of Demo World Assumptions
Random Ghost – Minimax Pacman
Why not minimax?

- Worst case reasoning is too conservative
- Need average case reasoning
Mixed Layer Types

- E.g. Backgammon
- Expectiminimax
  - Environment is an extra “random agent” player that moves after each min/max agent
  - Each node computes the appropriate combination of its children
Example: Backgammon

- Dice rolls increase $b$: 21 possible rolls with 2 dice
  - Backgammon $\approx$ 20 legal moves
  - Depth $2 = 20 \times (21 \times 20)^3 = 1.2 \times 10^9$

- As depth increases, probability of reaching a given search node shrinks
  - So usefulness of search is diminished
  - So limiting depth is less damaging
  - But pruning is trickier…

- Historic AI: TDGammon uses depth-2 search + very good evaluation function + reinforcement learning:
  world-champion level play

- 1st AI world champion in any game!
Utilities
Utilities

- Utilities are functions from outcomes (states of the world) to real numbers that describe an agent’s preferences.

- Where do utilities come from?
  - In a game, may be simple (+1/-1)
  - Utilities summarize the agent’s goals
  - Theorem: any “rational” preferences can be summarized as a utility function

- We hard-wire utilities and let behaviors emerge
  - Why don’t we let agents pick utilities?
  - Why don’t we prescribe behaviors?