Announcements

- Optional Mini-Contest (search)
  - due Friday 3/1 at 11:59p
  - Use Autolab for submissions

- Homework 3: Games
  - has been released, due 3/3 at 11:59p.
  - Few questions about Utilities (next lecture)

- Project 1
  - Please resubmit on Autolab today
  - single ZIP file (search.py and searchAgents.py only — no folders)

New Rules:

- Homework: submit answers on edX
  - individual work/individual submission
- Projects: submit source code on Autolab
  - team work/individual submission
- Contest: submit source code on Autolab
  - team work/team submission

All assignments will be due at 11:59pm EST
No late submissions
No submissions by e-mail

CS 188: Artificial Intelligence
Stochastic Games

Instructor: Marco Alvarez
University of Rhode Island

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.
All CS188 materials are available at http://ai.berkeley.edu.]

Recap - Adversarial Games

Minimax Example

Minimax Implementation

Minimax Efficiency

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:
  \[ f(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) \]
  - e.g. \( f(s) \) = (num white queens - num black queens), etc.

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
  - Guarantee of optimal play is gone
  - More plies makes a BIG difference
  - Use iterative deepening for an anytime algorithm

Why Pacman Starves

- 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!

Game Tree Pruning

Minimax Pruning

Alpha-Beta Implementation

- 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
  - With “perfect ordering”:
    - Time complexity drops to \( O(n^{d/2}) \)
    - Doubles solvable depth!
  - Full search of, e.g. chess, is still hopeless...
- This is a simple example of metareasoning (computing about what to compute)

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
  - 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

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-value(state)
    if the next agent is EXP: return exp-value(state)

def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v

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

Expectimax Example

\[
v = \frac{1}{2} \times 8 + \frac{1}{3} \times 24 + \frac{1}{6} \times (-12) = 10
\]

Expectimax Pruning?

Depth-Limited Expectimax

Estimate the value of an optimal path may require a lot of work to compute

Probabilities
Reminder: 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: \( P(T=\text{none}) = 0.25, P(T=\text{light}) = 0.50, P(T=\text{heavy}) = 0.25 \).

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:
- \( P(T=\text{heavy}) = 0.25 \), \( P(T=\text{heavy} \mid \text{Hour}=8\text{am}) = 0.60 \).

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?
- Time: 20 min, 30 min, 60 min.
- Probability: 0.25, 0.50, 0.25.
- Expected time: \( 20 \times 0.25 + 30 \times 0.50 + 60 \times 0.25 = 35 \text{ min} \).

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.

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, which has the nice property that it all collapses into one game tree.

Assumptions vs. Reality

<table>
<thead>
<tr>
<th></th>
<th>Adversarial Ghost</th>
<th>Random Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimax Pacman</td>
<td>Won 5/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Avg. Score</td>
<td>483</td>
<td>493</td>
</tr>
<tr>
<td>Expectimax Pacman</td>
<td>Won 1/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Avg. Score</td>
<td>-120</td>
<td>503</td>
</tr>
</tbody>
</table>

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.

Modeling Assumptions

The Dangers of Optimism and Pessimism

Dangerous Optimism
- Assuming chance when the world is adversarial.
- Assumes the worst case when it's not likely.

Dangerous Pessimism
- Assuming the worst case when it's not likely.

Video of Demo World Assumptions

Random Ghost – Expectimax Pacman

Adversarial Ghost – Minimax Pacman

Results from playing 5 games.

[Demos: world assumptions (L7D3,4,5,6)]
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

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...

| 1,7 | 5,2 | 6,7 | 2,7 |
| 5,7 | 1,5 | 7,1 | 2,6 |