CSE 473: Introduction to Artificial Intelligence

Hanna Hajishirzi
Expectimax – Complex Games

slides adapted from
Dan Klein, Pieter Abbeel ai.berkeley.edu
And Dan Weld, Luke Zettlemoyer
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**
Video of Demo Min vs. Exp (Min)

SCORE: 0

SCORE: 0
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 max-value(state):
    initialize \( v = -\infty \)
    for each successor of state:
        \( v = \max(v, value(successor)) \)
    return v

def exp-value(state):
    initialize \( v = 0 \)
    for each successor of state:
        \( p = probability(successor) \)
        \( v += p \times value(successor) \)
    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 Pruning?
Depth-Limited Expectimax

Estimate of true expectimax value (which would require a lot of work to compute)
CSE 473: Introduction to Artificial Intelligence

Hanna Hajishirzi
Expectimax – Complex Games

slides adapted from
Dan Klein, Pieter Abbeel ai.berkeley.edu
And Dan Weld, Luke Zettlemoyer
Expectimax Pseudocode

```python
def exp_value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

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

Estimate of true expectimax value (which would 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:
- \( P(T=\text{heavy}) = 0.25, P(T=\text{heavy} \mid \text{Hour=8am}) = 0.60 \)
- 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>
</tr>
</thead>
<tbody>
<tr>
<td>20 min</td>
<td>0.25</td>
</tr>
<tr>
<td>30 min</td>
<td>0.50</td>
</tr>
<tr>
<td>60 min</td>
<td>0.25</td>
</tr>
</tbody>
</table>

  \[ 20 \text{ min} \times 0.25 + 30 \text{ min} \times 0.50 + 60 \text{ min} \times 0.25 = 35 \text{ min} \]
In expectimax search, we have a probabilistic model of how the opponent (or environment) 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.
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
Assumptions vs. Reality

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>

Pacman used depth 4 search with an eval function that avoids trouble
Ghost used depth 2 search with an eval function that seeks Pacman
Video of Demo World Assumptions
Random Ghost – Expectimax Pacman
Video of Demo World Assumptions
Adversarial Ghost – Minimax Pacman
Video of Demo World Assumptions
Random Ghost – Minimax Pacman
Video of Demo World Assumptions
Adversarial Ghost – Expectimax Pacman
Assumptions vs. Reality

Results from playing 5 games

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Pacman Score</th>
<th>Ghost Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimax</td>
<td>Won 5/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td></td>
<td>Avg. Score: 483</td>
<td>Avg. Score: 493</td>
</tr>
<tr>
<td>Expectimax</td>
<td>Won 1/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td></td>
<td>Avg. Score: -303</td>
<td>Avg. Score: 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.
Why not minimax?

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

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

\[
\begin{align*}
\text{if state is a Max node then} & \quad \text{return the highest ExpectiMinimax-Value of Successors(state)} \\
\text{if state is a Min node then} & \quad \text{return the lowest ExpectiMinimax-Value of Successors(state)} \\
\text{if state is a chance node then} & \quad \text{return average of ExpectiMinimax-Value of Successors(state)}
\end{align*}
\]
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!
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…
Utilities

- Utilities: values that we assign to every state

- Why should we average utilities? Why not minimax?

- Principle of maximum expected utility:
  - A rational agent should choose the action that maximizes its expected utility, given its knowledge
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

- We hard-wire utilities and let behaviors emerge.
Utilities: Uncertain Outcomes

Getting ice cream

Get Single

Get Double

Oops

Whew!
What Utilities to Use?

- For worst-case minimax reasoning, terminal function scale doesn’t matter
  - We just want better states to have higher evaluations (get the ordering right)

- For average-case expectimax reasoning, we need *magnitudes* to be meaningful
Review and Next Topics

- **Adversarial Games**
  - Minimax search
  - $\alpha$-$\beta$ search
  - Evaluation functions
  - Multi-player, non-0-sum

- **Stochastic Games**
  - Expectimax
  - Markov Decision Processes
  - Reinforcement Learning