Worst-Case vs. Average Case

New type of node!

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

```python
def max_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)
```

```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 EXP:
        return exp_value(state)
```

```python
def exp_value(state):
    initialize v = 0
    for each successor of state:
        v = maxv, value(successor)
        return v
```
**Expectimax Pseudocode**

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

**Expectimax Example**

```
10
\[\begin{array}{c}
1/2 & 2/3 & 1/6 \\
8 & 26 & 12
\end{array}\]
```

- Expectimax Pruning?

```
10
\[\begin{array}{c}
1/2 & 2/3 & 1/6 \\
8 & 26 & 12
\end{array}\]
```

- **Depth-Limited Expectimax**

```
10
\[\begin{array}{c}
1/2 & 2/3 & 1/6 \\
8 & 26 & 12
\end{array}\]
```

- **Probabilities**

```
0.25
0.50
0.25
```

- **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
  - 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>
</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>

35 min

What Probabilities to Use?

- In expectimax search, we have a probability 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
  - 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

Informed Probabilities

- Let’s say you know that your opponent is sometimes lazy. 20% of the time, she moves randomly, but usually (80%) she runs a depth 2 minimax to decide her move
- 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

Modeling Assumptions

The Dangers of Optimism and Pessimism

<table>
<thead>
<tr>
<th>Dangerous Optimism</th>
<th>Dangerous Pessimism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assuming chance when the world is adversarial</td>
<td>Assuming the worst case when it’s not likely</td>
</tr>
</tbody>
</table>

Video of Demo World Assumptions

Random Ghost – Expectimax Pacman
### Video of Demo World Assumptions

**Adversarial Ghost – Minimax Pacman**

- [Image of Minimax Pacman](#)

**Adversarial Ghost – Expectimax Pacman**

- [Image of Expectimax Pacman](#)

### Video of Demo World Assumptions

**Random Ghost – Minimax Pacman**

- [Image of Random Ghost Minimax Pacman](#)

### Assumptions vs. Reality

<table>
<thead>
<tr>
<th></th>
<th>Adversarial Ghost</th>
<th>Random Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimax Pacman</strong></td>
<td>Won 5/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td><strong>Avg. Score</strong></td>
<td>483</td>
<td>493</td>
</tr>
<tr>
<td><strong>Expectimax Pacman</strong></td>
<td>Won 1/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td><strong>Avg. Score</strong></td>
<td>-303</td>
<td>503</td>
</tr>
</tbody>
</table>

Results from playing 5 games.

Pacman used depth 4 search with an eval function that avoids trouble.

Ghost used depth 2 search with an eval function that seeks Pacman.

### Other Game Types

- **Example: Backgammon**

- [Image of Backgammon](#)
Mixed Layer Types

- E.g. Backgammon
- Expectminimax
- 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 = 20 legal moves
- Depth 2 = 20 x (2 x 20)^2 = 1.2 x 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 (1992): TDGammon uses depth-2 search + very good evaluation function + reinforcement learning: world-champion level play
- 1st AI world champion in any game!

Different Types of Ghosts?

- Stupid
- Devilish Smart

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

- Why should we average utilities?
- Principle of maximum expected utility:
  - A rational agent should choose the action that maximizes its expected utility, given its knowledge
- Questions:
  - Where do utilities come from?
  - How do we know such utilities even exist?
  - How do we know that averaging even makes sense?
  - What if our behavior (preferences) can't be described by utilities?
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)
- We call this insensitivity to monotonic transformations
- For average-case expectimax reasoning, we need magnitudes to be meaningful

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?

Utilities: Uncertain Outcomes

Preferences

- An agent must have preferences among:
  - Prizes: A, B, etc.
  - Lotteries: situations with uncertain prizes

- Notation:
  - Preference: $A \succ B$
  - Indifference: $A \sim B$

Rationality

Rational Preferences

- We want some constraints on preferences before we call them rational, such as:

  Axiom of Transitivity: $(A \succ B) \land (B \succ C) \Rightarrow (A \succ C)$

- For example: an agent with intransitive preferences can be induced to give away all of its money
  - If $B \succ C$, then an agent with $C$ would pay (say) 1 cent to get $B$
  - If $A \succ B$, then an agent with $B$ would pay (say) 1 cent to get $A$
  - If $C \succ A$, then an agent with $A$ would pay (say) 1 cent to get $C$
The Axioms of Rationality

- Transitivity: \((A > B) \land (B > C) \Rightarrow (A > C)\)
- Continuity: \(A > B > C \Rightarrow \exists \alpha \in [a, b, c] \) \(\sim A - \alpha \sim C \)
- Subjectivity: \(A \sim B \Rightarrow [b - a] \sim [c - a] \)
- Monotonicity: \(A > B \Rightarrow 1_A > 1_B \)

Theorem: Rational preferences imply behavior describable as maximization of expected utility.

The MEU Principle

- Given any preferences satisfying these constraints, there exists a real-valued function \(U\) such that:
  \[ U(A) \geq U(B) \iff A \sim B \]
  \[ U(p_1 S_1 + \ldots + p_n S_n) = \sum p_i U(S_i) \]

  i.e., values assigned by \(U\) preserve preferences of both prizes and lotteries!

- Maximum expected utility (MEU) principle:
  - Choose the action that maximizes expected utility.
  - Note: an agent can be entirely rational (consistent with MEU) without ever representing or manipulating utilities and probabilities.
  - e.g., a lookup table for perfect tic-tac-toe, a reflex vacuum cleaner.

Human Utilities

- Normalized utilities: \(u_\text{max} = 1.0, \ u_\text{min} = 0.0\)
- Micromorts: one-millionth chance of death, useful for paying to reduce product risks, etc.
- QALYs: quality-adjusted life years, useful for medical decisions involving substantial risk.
- Note: behavior is invariant under positive linear transformation.
  \[ U'(x) = k_1 U(x) + k_2 \quad \text{where } k_1 > 0 \]
- With deterministic prizes only (no lottery choices), only ordinal utility can be determined, i.e., total order on prizes.

Money

- Money does not behave as a utility function, but we can talk about the utility of having money (or being in debt).
- Given a lottery \(L = [p, X; 1 - p, Y]\),
  - The expected monetary value \(EMV(L) = pX + (1-p)Y\).
  - \(U(L) = p^*U(X) + (1-p^*)U(Y)\)
  - Typically, \(U(L) < U(EMV(L))\).
  - In this sense, people are risk-averse.
  - When deep in debt, people are risk-prone.
Example: Insurance

- Consider the lottery [0.5, $1000; 0.5, $0]
  - What is its expected monetary value? ($500)
  - What is its certainty equivalent?
    - Monetary value acceptable in lieu of lottery
    - $400 for most people
  - Difference of $100 is the insurance premium
    - There’s an insurance industry because people
      will pay to reduce their risk
    - If everyone were risk-neutral, no insurance
      needed.
  - It’s win-win: you’d rather have the $400 and
    the insurance company would rather have the
    lottery (their utility curve is flat and they have
    many lotteries).

Example: Human Rationality?

- Famous example of Allais (1953)
  - A: [0.8, $4k; 0.2, $0]
  - B: [1.0, $3k; 0.0, $0]
  - C: [0.2, $4k; 0.8, $0]
  - D: [0.25, $3k; 0.75, $0]
- Most people prefer B > A, C > D
- But if U($0) = 0, then
  - B > A \( \Rightarrow U($3k) > 0.8 U($4k) \)
  - C > D \( \Rightarrow 0.8 U($4k) > U($3k) \)