Expectimax Search Trees

- What if we don’t know what the result of an action will be? E.g.,
  - In solitaire, next card is unknown
  - In minesweeper, mine locations
  - In pacman, the ghosts act randomly
- Can do expectimax search to maximize average score
  - Chance nodes, like min nodes, except the outcome is uncertain
  - Calculate expected utilities
  - Max nodes as in minimax search
  - Chance nodes take average (expectation) of value of children
- Later, we’ll learn how to formalize the underlying problem as a Markov Decision Process

Expectimax Pseudocode

```python
def value(s):
    if s is a max node return maxValue(s)
    if s is an exp node return expValue(s)
    if s is a terminal node return evaluation(s)

def maxValue(s):
    values = [value(s') for s' in successors(s)]
    return max(values)

def expValue(s):
    values = [value(s') for s' in successors(s)]
    weights = [probability(s, s') for s' in successors(s)]
    return expectation(values, weights)
```

Expectimax Quantities

Expectimax Pruning?

Expectimax Search

- Chance nodes
  - Chance nodes are like min nodes, except the outcome is uncertain
  - Calculate expected utilities
  - Chance nodes average successor values (weighted)
- Each chance node has a probability distribution over its outcomes (called a model)
  - For now, assume we’re given the model
- Utilities for terminal states
  - Static evaluation functions give us limited-depth search

Expectimax Search Trees

- Estimate of true expectimax value (which would require a lot of work to compute)
Expectimax for Pacman

- Notice that we’ve gotten away from thinking that the ghosts are trying to minimize pacman’s score.
- Instead, they are now a part of the environment.
- Pacman has a belief (distribution) over how they will act.
- Quiz: Can we see minimax as a special case of expectimax?
- Quiz: what would pacman’s computation look like if we assumed that the ghosts were doing 1-ply minimax and taking the result 20% of the time, otherwise moving randomly?
- If you take this further, you end up calculating belief distributions over your opponents’ belief distributions over your belief distributions, etc…
  - Can get unmanageable very quickly!

Expectimax Utilities

- For minimax, 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 expectimax, we need magnitudes to be meaningful

Maximum Expected Utility

- Why should we average utilities? Why not minimax?
- Principle of maximum expected utility:
  - A rational agent should choose the action which maximizes its expected utility, given its knowledge.
- Questions:
  - Where do utilities come from?
  - How do we know such utilities even exist?
  - Why are we taking expectations of utilities (not, e.g. minimax)?
  - What if our behavior can’t be described by 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?

Utilities: Uncertain Outcomes

- Going to airport from home
  - Get Double
  - Get Single

Results from playing 5 games

<table>
<thead>
<tr>
<th></th>
<th>Minimizing Ghost</th>
<th>Random Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimax</td>
<td>Won 5/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Pacman</td>
<td>Avg. Score: 493</td>
<td>Avg. Score: 483</td>
</tr>
<tr>
<td>Expectimax</td>
<td>Won 1/5</td>
<td>Won 5/5</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.
Preferences

- An agent chooses among:
  - Prizes: A, B, etc.
  - Lotteries: situations with uncertain prizes
    \[ L = [p, A; (1 - p), B] \]
- Notation:
  - \( A > B \): A preferred over B
  - \( A \sim B \): indifference between A and B
  - \( A \geq B \): B not preferred over A

Rational Preferences

- We want some constraints on preferences before we call them rational
  \[ (A > B) \land (B > C) \implies (A > C) \]
- For example: an agent with intransitive preferences can be induced to give away all of its money
  - If \( B > C \), then an agent with \( C \) would pay (say) 1 cent to get B
  - If \( A > B \), then an agent with \( B \) would pay (say) 1 cent to get A
  - If \( C > A \), then an agent with \( A \) would pay (say) 1 cent to get C

Rational Preferences

- Preferences of a rational agent must obey constraints.
  - The axioms of rationality:
    - Orderability
      \[ (A > B) \lor (B > A) \lor (A \equiv B) \]
    - Transitivity
      \[ (A > B) \land (B > C) \implies (A > C) \]
    - Continuity
      \[ A > B > C \implies \exists p \in [0, 1] \text{ s.t. } [p, A; (1 - p), C] \sim B \]
    - Substitutability
      \[ A \sim B \implies [p, A; (1 - p), B] \sim [p, B; (1 - p), C] \]
    - Monotonicity
      \[ A > B \implies [p, A; (1 - p), B] \geq [p, A; (1 - p), B] \]
- Theorem: Rational preferences imply behavior describable as maximization of expected utility

Utility Scales

- Normalized utilities: \( u_0 = 1.0, u_0 = 0.0 \)
- 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
  \[ U'(x) = k_1 U(x) + k_2 \] where \( k_1 > 0 \)
- With deterministic prizes only (no lottery choices), only ordinal utility can be determined, i.e., total order on prizes

MEU Principle

- Theorem:
  \[ \{ \text{Ramsey, 1931; von Neumann & Morgenstern, 1944} \} \]
  Given any preferences satisfying these constraints, there exists a real-valued function \( U \) such that:
  \[ U(A) \geq U(B) \iff A \geq B \]
  \[ U([p_1, S_1; \ldots; p_n, S_n]) = \sum p_i U(S_i) \]
- Maximum expected likelihood (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 tictactoe, reflex vacuum cleaner

Human Utilities

- Utilities map states to real numbers. Which numbers?
- Standard approach to assessment of human utilities:
  - Compare a state \( A \) to a standard lottery \( L_p \) between
  - “best possible prize” \( u_0 \) with probability \( p \)
  - “worst possible catastrophe” \( u_0 \) with probability \( 1 - p \)
  - Adjust lottery probability \( p \) until \( A \equiv L_p \)
  - Resulting \( p \) is a utility in \([0, 1] \)

pay $30 \sim 0.999999$  instant death

\[ \begin{array}{c}
\text{Continue as before} \\
0.999999 \\
0.000001 \\
\text{Instant death}
\end{array} \]
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 \( EMV(L) = p*X + (1-p)*Y \)
  - Typically, \( U(L) < U(EMV(L)) \): why?
- In this sense, people are risk-averse
- When deep in debt, we are risk-prone

Utility curve: for what probability \( p \) am I indifferent between:
- Some sure outcome \( x \)
- A lottery \([p,\$M; (1-p),\$0]\), \( M \) large

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!

Example: Insurance

- Because people ascribe different utilities to different amounts of money, insurance agreements can increase both parties’ expected utility

You own a car. Your lottery:
\[ L_Y = [0.8, \$0; 0.2, -$200] \]
i.e., 20% chance of crashing

You do not want -$200!

\[ U_Y(L_Y) = 0.2*U_Y(-$200) = -200 \]
\[ U_Y(-$200) = -150 \]

<table>
<thead>
<tr>
<th>Amount</th>
<th>Your Utility ( U_Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0</td>
<td>0</td>
</tr>
<tr>
<td>-$50</td>
<td>-150</td>
</tr>
<tr>
<td>-$200</td>
<td>-1000</td>
</tr>
</tbody>
</table>

Insurance company buys risk:
\[ L_I = [0.8, \$50; 0.2, -$150] \]
i.e., \$50 revenue + your \( L_Y \)

Insurer is risk-neutral:

\[ U(L_I) = U(0.8*50 + 0.2*(-150)) \]
\[ = U(10) > U(0) \]

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
  - \( B > A \Rightarrow U(\$3k) > 0.8 U(\$4k) \)
  - \( C > D \Rightarrow 0.8 U(\$4k) > U(\$3k) \)

Non-Zero-Sum Utilities

- Similar to minimax:
  - Terminals have utility tuples
  - Node values are also utility tuples
  - Each player maximizes its own utility
  - Can give rise to cooperation and competition dynamically...
Mixed Layer Types

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

Expectiminimax-Value(state):
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)

Stochastic Two-Player

- Dice rolls increase \( b \): 21 possible rolls with 2 dice
- Backgammon \( \approx 20 \) legal moves
- Depth 2 \( = 20 \times (21 \times 20)^2 \approx 1.2 \times 10^9 \)
- 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...
- TDGammon uses depth-2 search + very good evaluation function + reinforcement learning:
  world-champion level play
- 1st AI world champion in any game!