Minimax Example

Suicidal agent
Expectimax

- Uncertain outcomes are controlled by chance not an adversary
- Chance nodes are new types of nodes (instead of Min nodes)
Worst-case vs. Average Case

- 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
Stochastic Single-Player

- What if we don’t know what the result of an action will be? E.g.,
  - In solitaire, shuffle is unknown
  - In minesweeper, mine locations

- Can do **expectimax search**
  - Chance nodes, like actions except the environment controls the action chosen
  - Max nodes as before
  - Chance nodes take average (expectation) of value of children
Expectimax Pseudocode

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

\[
v = \left(\frac{1}{2}\right)(8) + \left(\frac{1}{3}\right)(24) + \left(\frac{1}{6}\right)(-12) = 10\]
Maximum Expected Utility

- Why should we average utilities? Why not minimax?

- Principle of maximum expected utility: an agent should choose the action which maximizes its expected utility, given its knowledge
  - General principle for decision making
  - Often taken as the definition of rationality
  - We’ll see this idea over and over in this course!

- Let’s decompress this definition…
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: $P(T=\text{none}) = 0.25$, $P(T=\text{light}) = 0.55$, $P(T=\text{heavy}) = 0.20$

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:
- $P(T=\text{heavy}) = 0.20$, $P(T=\text{heavy} \mid \text{Hour}=8\text{am}) = 0.60$
- We’ll talk about methods for reasoning and updating probabilities later
What are Probabilities?

- **Objectivist / frequentist answer:**
  - Averages over repeated experiments
  - E.g. empirically estimating $P(\text{rain})$ from historical observation
  - E.g. pacman’s estimate of what the ghost will do, given what it has done in the past
  - Assertion about how future experiments will go (in the limit)
  - Makes one think of inherently random events, like rolling dice

- **Subjectivist / Bayesian answer:**
  - Degrees of belief about unobserved variables
  - E.g. an agent’s belief that it’s raining, given the temperature
  - E.g. pacman’s belief that the ghost will turn left, given the state
  - Often *learn* probabilities from past experiences (more later)
  - New evidence *updates beliefs* (more later)
Uncertainty Everywhere

- Not just for games of chance!
  - I’m sick: will I sneeze this minute?
  - Email contains “FREE!”: is it spam?
  - Tooth hurts: have cavity?
  - 60 min enough to get to the airport?
  - Robot rotated wheel three times, how far did it advance?
  - Safe to cross street? (Look both ways!)

- Sources of uncertainty in random variables:
  - Inherently random process (dice, etc)
  - Insufficient or weak evidence
  - Ignorance of underlying processes
  - Unmodeled variables
  - The world’s just noisy – it doesn’t behave according to plan!
Reminder: Expectations

- We can define function f(X) of a random variable X

- The expected value of a function is its average value, weighted by the probability distribution over inputs

- Example: How long to get to the airport?
  - Length of driving time as a function of traffic:
    \[ L(\text{none}) = 20, \ L(\text{light}) = 30, \ L(\text{heavy}) = 60 \]
  - What is my expected driving time?
    - Notation: \( E_{P(T)}[L(T)] \)
    - Remember, \( P(T) = \{\text{none: 0.25, light: 0.5, heavy: 0.25}\} \)

\[
E[ L(T) ] = L(\text{none}) \times P(\text{none}) + L(\text{light}) \times P(\text{light}) + L(\text{heavy}) \times P(\text{heavy})
\]
\[
E[ L(T) ] = (20 \times 0.25) + (30 \times 0.5) + (60 \times 0.25) = 35
\]
Review: Expectations

- Real valued functions of random variables:
  \[ f : X \rightarrow R \]

- Expectation of a function of a random variable
  \[ E_{P(X)}[f(X)] = \sum_x f(x)P(x) \]

- Example: Expected value of a fair die roll

<table>
<thead>
<tr>
<th>X</th>
<th>P</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1/6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1/6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1/6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1/6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>1/6</td>
<td>6</td>
</tr>
</tbody>
</table>

\[ 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + 3 \times \frac{1}{6} + 4 \times \frac{1}{6} + 5 \times \frac{1}{6} + 6 \times \frac{1}{6} \]

\[ = 3.5 \]
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 set of preferences between outcomes can be summarized as a utility function (provided the preferences meet certain conditions)

In general, we hard-wire utilities and let actions emerge

More on utilities soon…
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**
  - 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**
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 node for every outcome out of our control: opponent or environment
- The model might say that adversarial actions are likely!

For now, assume for any state we magically have a distribution to assign probabilities to opponent actions / environment outcomes.
Expectimax Pruning

![Expectimax Pruning Diagram]

- Expectimax Pruning
- Expectimax Example
- Node Values: 3, 12, 9, 2, 4, 6, 15, 6, 0
Expectimax Pruning

- Not easy
  - exact: need bounds on possible values
  - approximate: sample high-probability branches
Depth-limited Expectimax

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

- Evaluation functions quickly return an estimate for a node’s true value (which value, expectimax or minimax?)
- For minimax, evaluation 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

![Diagram showing evaluation functions with different scales and magnitudes]
Expectediminax 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 expectiminax?
Quiz

- 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
### Expectimax for Pacman

#### Results from playing 5 games

<table>
<thead>
<tr>
<th></th>
<th>Minimizing 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></td>
<td>Avg. Score: 493</td>
<td>Avg. Score: 483</td>
</tr>
<tr>
<td><strong>Expectimax Pacman</strong></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 does depth 4 search with an eval function that avoids trouble
Minimizing ghost does depth 2 search with an eval function that seeks Pacman
Mixed Layer Types

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

```plaintext
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 4 = $20 \times (21 \times 20)^3 = 1.2 \times 10^9$
- As depth increases, probability of reaching a given node shrinks
  - So value of lookahead is diminished
  - So limiting depth is less damaging
  - But pruning is less possible…
- TDGammon uses depth-2 search + very good eval function + reinforcement learning: world-champion level play
Multi-player Non-Zero-Sum Games

- Similar to minimax:
  - Utilities are now tuples
  - Each player maximizes their own entry at each node
  - Propagate (or back up) nodes from children
  - Can give rise to cooperation and competition dynamically…