Announcements

- No section on Monday! Enjoy the break
- Check website for extra office hours next week
- Written Assignment 1:
  - Solutions posted today
  - Grading has commenced; to be returned in section
- Project 2:
  - Due next Wednesday (a week from yesterday)
- New bSpace forum for general comments
- You can call me John
- Warning: we’re jumping ahead in the textbook

Logic, Reasoning & Planning

- Huge subfield with a long history in AI
- Knowledge-based agents reason with what they know
- Logical reasoning
  \[ \forall x \text{ King } (x) \Rightarrow \text{ Person } (x) \]
  \[ \text{King } (\text{Richard}) \]
  \[ \text{Person } (\text{Richard}) \]
- Knowledge representation
  \[ \forall m, c \text{ Mother } (c) = m \Leftrightarrow \text{ Female } (m) \land \text{ Parent } (m, c) \]
- Planning

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

Today

- Expectimax wrap-up
- Utility theory
- Reinforcement learning

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
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)
```

Non-Zero-Sum Games

- Similar to max or minimax:
- Terminal states have utility tuples
- Each player maximizes its own utility
- Value of a node is a whole tuple
- Can give rise to cooperation or competition dynamically...

Stochastic Two-Player

- Dice rolls increase $b$: 21 possible rolls with 2 dice
  - Backgammon = 20 legal moves
  - Depth $2^2 = 20 \times (21 \times 20) = 1.2 \times 10^9$
- 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!

Maximizing Expected Utility

- Principle of maximum expected utility:
  - A rational agent should choose the action that maximizes its expected utility, given its knowledge
- Where do utilities come from?
- How do we know such utilities even exist?
- Why are we taking expectations of utilities?
- What if our behavior can't be described by utilities?

Preferences

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

Rational Preferences

- We want some constraints on preferences before we call them rational
  \[(A \succ B) \land (B \succ C) \Rightarrow (A \succ C)\]
- 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 $C$
Rational Preferences

- Preferences of a rational agent must obey constraints.
- The axioms of rationality:
  - Orderability: \((A > B) \lor (B > A) \lor (A \sim B)\)
  - Transitivity: \((A > B) \land (B > C) \Rightarrow (A > C)\)
  - Continuity: \(A > B > C \Rightarrow \exists p \in [0, 1] \text{ s.t. } pA + (1 - p)C > B\)

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

MEU Principle

- Theorem:
  - [Ramsey, 1931; von Neumann & Morgenstern, 1944]
  - Given any preferences satisfying these axioms, there exists a real-valued function \(U\) such that:
    - 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 – this is about behavior
      - E.g., a lookup table for perfect tic-tac-toe

Utility Scales

- Normalized utilities: \(u_+ = 1.0, u_- = 0.0\)
- Micromorts: one-millionth chance of death, useful for paying to reduce product risks, etc. Worth about $20 in 1980.
- 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 \(EMV(L) = pX + (1-p)Y\)
  - Typically, \(U(L) < U(EMV(L))\): people prefer a sure thing
- When deep in debt (or in Vegas), we are risk-prone

Example: Insurance

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

  John owns a car. His lottery:
  \(L_J = [0.8, 0; 0.2, -200]\)
  i.e., 20% chance of crashing

  John is risk-averse. He does not want $200!

  \(U_J(L_J) = 0.2^*U_J(-200) = -200\)
  \(U_J(-$50) = -150\)

  Amount | John’s Utility
  --- | ---
  $0 | 0
  -$200 | -1000
  -$40 | -100
  -$50 | -150

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

  John is risk-averse. He does not want $200!

  \(U_I(L_I) = 0.2^*U_I(-200) = -200\)
  \(U_I(-$50) = -150\)

  Insurance is risk-neutral:
  \(U_I(L_I) = U_I(EMV(L_I))\)

  \(U_I(L_I) = U_I(0.8^*50 + 0.2^*(-150)) = U_I(-10) > U_I(0)\)
Are Humans Rational?

- 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]

Famous example of Allais (1953)
Most people prefer B > A, C > D
But if U($0) = 0, then
- B > A ⇒ U($3k) > 0.8 U($4k)
- C > D ⇒ 0.8 U($4k) > U($3k)

One explanation: people don't want to feel regret.

Reinforcement Learning

- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must learn to act so as to maximize expected rewards
  - Change the rewards, change the learned behavior

- Examples:
  - Playing a game, reward at the end for winning / losing
  - Vacuuming a house, reward for each piece of dirt picked up
  - Automated taxi, reward for each passenger delivered
  - [Crawler demo]

Grid World

- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Big rewards come at the end

Markov Decision Processes

- An MDP is defined by:
  - A set of states \( s \subseteq S \)
  - A set of actions \( a \subseteq A \)
  - A transition function \( T(s, a, s') \)
    - Prob that a from s leads to s'
    - i.e., \( P(s' | s, a) \)
    - Also called the model
  - A reward function \( R(s, a, s') \)
    - Sometimes just \( R(s) \) or \( R(s') \)
  - A start state (or distribution)
  - Maybe a terminal state

- MDPs are a family of non-deterministic search problems
  - Reinforcement learning: MDPs where we don't know the transition or reward functions

Solving MDPs

- In deterministic single-agent search problem, want an optimal plan, or sequence of actions, from start to a goal
- In an MDP, we want an optimal policy \( \pi^*: S \rightarrow A \)
  - A policy \( \pi \) gives an action for each state
  - An optimal policy maximizes expected utility if followed
  - Defines a reflex agent

Example Optimal Policies

- Optimal policy when \( R(s, a, s') = -0.02 \) for all non-terminals s

<table>
<thead>
<tr>
<th>State</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.001</td>
</tr>
<tr>
<td>2</td>
<td>-0.02</td>
</tr>
<tr>
<td>3</td>
<td>-0.4</td>
</tr>
<tr>
<td>4</td>
<td>-0.20</td>
</tr>
</tbody>
</table>
Example: High-Low
- Three card types: 2, 3, 4
- Infinite deck, twice as many 2's
- Start with 3 showing
- After each card, you predict the next card will be "high" or "low" compared to the current one
- New card is flipped
- If you're right, you win the points shown on the new card
- Ties are no-ops
- If you're wrong, game ends
- Differences from expectimax:
  - #1: get rewards as you go
  - #2: you might play forever!

High-Low as an MDP
- States: 2, 3, 4, done
- Actions: High, Low
- Model: \( T(s,a,s') \):
  - \( P(s'=\text{done} \mid 4, \text{High}) = 3/4 \)
  - \( P(s'=2 \mid 4, \text{High}) = 0 \)
  - \( P(s'=3 \mid 4, \text{High}) = 0 \)
  - \( P(s'=4 \mid 4, \text{High}) = 1/4 \)
  - \( P(s'=\text{done} \mid 4, \text{Low}) = 0 \)
  - \( P(s'=2 \mid 4, \text{Low}) = 1/2 \)
  - \( P(s'=3 \mid 4, \text{Low}) = 1/4 \)
  - \( P(s'=4 \mid 4, \text{Low}) = 1/4 \)
- Rewards: \( R(s,a,s') \):
  - Number shown on \( s' \) if \( s \neq s' \)
  - 0 otherwise
- Start: 3

High-Low

MDP Search Trees
- Each MDP state gives an expectimax-like search tree

What's Coming Up in MDPs?
- Maximizing expected utility in MDPs
- Finding optimal policies
- Choosing policies when the transition model is unknown
- Generalizing over similar states
- Creating smart reflex agents
- Pacman agents that learn how to play well