Introduction to Decision Making

CS 486/686: Introduction to Artificial Intelligence
Outline

• Utility Theory
• Decision Trees
Decision Making Under Uncertainty

• I give a robot a planning problem: “I want coffee”
  - But the coffee maker is broken: Robot reports “No plan!”
Decision Making Under Uncertainty

• I want more robust behaviour
• I want my robot to know what to do when my primary goal is not satisfied
  - Provide it with some indication of my preferences over alternatives
    - e.g. coffee better than tea, tea better than water, water better than nothing,...
Decision Making Under Uncertainty

• But it is more complicated than that
  - It could wait 45 minutes for the coffee maker to be fixed

• What is better?
  - Tea now?
  - Coffee in 45 minutes?
Preferences

• A preference ordering $\succeq$ is a ranking over all possible states of the world $s$

• These could be outcomes of actions, truth assignments, states in a search problem, etc

- $s \succeq t$: state $s$ is at least as good as state $t$
- $s > t$: state $s$ is strictly preferred to state $t$
- $s \sim t$: agent is ambivalent between states $s$ and $t$
Preferences

• If an agent’s actions are deterministic, then we know what states will occur.
• If an agent’s actions are not deterministic, then we represent this by lotteries.
  - Probability distribution over outcomes
  - Lottery $L=[p_1,s_1;p_2,s_2;...;p_n,s_n]$ where $s_1$ occurs with probability $p_1$, $s_2$ occurs with probability $p_2$, ...
• Orderability: Given 2 states A and B
  - \((A \gneq B) \lor (B \gneq A) \lor (A \sim B)\)

• Transitivity: Given 3 states A, B, C
  - \((A \gneq B) \land (B \gneq C) \rightarrow (A \gneq C)\)

• Continuity:
  - \(A \gneq B \geq C \rightarrow \text{Exists } p, [p, A; (1-p), C] \sim B\)

• Substitutability
  - \(A \sim B \rightarrow [p, A; 1-p, C] \sim [p, B, 1-p, C]\)

• Monotonicity:
  - \((A \gneq B) \rightarrow (p \geq q \leftrightarrow [p, A; 1-p, B] \geq [q, A; 1-q, B])\)

• Decomposability
  - \([p, A; 1-p[q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]\)
Why Impose These Conditions?

• Structure of preference ordering imposes certain “rationality requirements”
  
  - It is a weak ordering

• Example: Why transitivity?
Money Pump

A>B>C>A
Decision Problem: Certainty

- **A decision problem under certainty** is <$D, S, f, ≿>$ where
  - $D$ is a set of decisions
  - $S$ is a set of outcomes or states
  - $f$ is an outcome function $f:D \rightarrow S$
  - $≿$ is a preference ordering over $S$

- **A solution** to a decision problem is any $d^*$ in $D$ such that $f(d^*)≿f(d)$ for all $d$ in $D$
Computational Issues

• At some level, a solution to a decision problem is trivial
  - But decisions and outcome functions are rarely specified explicitly
  - For example: In search you construct the set of decisions by exploring search paths
    - Do not know the outcomes in advance

Preferences
  - $c, b, bc$
  - $c, b, \sim bc$
  - $c, \sim b, \sim bc$
  - $c, \sim b, bc$
• Suppose actions do not have deterministic outcomes
  - Example: When the robot pours coffee, 20% of the time it spills it, making a mess
  - Preferences: $c, \neg \text{mess} > \neg c, \neg \text{mess} > \neg c, \text{mess}$

• What should your robot do?
  - Decision \textit{getcoffee} leads to a good outcome and a bad outcome with some probability
  - Decision \textit{donothing} leads to a medium outcome
Utilities

• Rather than just ranking outcomes, we need to quantify our degree of preference
  - How much more we prefer one outcome to another (e.g. c to \sim mess)

• A utility function $U: S \rightarrow \mathbb{R}$ associates a real-valued utility to each outcome
  - Utility measures your degree of preference for $s$

• $U$ induces a preference ordering $\succeq_U$ over $S$ where $s \succeq_U t$ if and only if $U(s) \geq U(t)$
Expected Utility

• Under conditions of uncertainty, decision d induces a distribution over possible outcomes

  - \( P_d(s) \) is the probability of outcome s under decision d

• The **expected utility** of decision d is

\[
EU(d) = \sum_{s \in S} P_d(s)U(s)
\]
• When my robot pours coffee, it makes a mess 20% of the time

• If $U(c, \neg ms)=10$, $U(\neg c, \neg ms)=5$, $U(\neg c, ms)=0$ then
  - $EU(getcoffee)=(0.8)10+(0.2)0=8$
  - $EU(donthing)=5$

• If $U(c, \neg ms)=10$, $U(\neg c, \neg ms)=9$, $U(\neg c, ms)=0$ then
  - $EU(getcoffee)=8$
  - $EU(donthing)=9$
Maximum Expected Utility Principle

• Principle of Maximum Expected Utility
  - The optimal decision under conditions of uncertainty is that with the greatest expected utility

• Robot example:
  - First case: optimal decision is getcoffee
  - Second case: optimal decision is donothing
Decision Problem: Uncertainty

• A decision problem under uncertainty is \(<D,S,P,U>\)
  - Set of decisions \(D\)
  - Set of outcomes \(S\)
  - Outcome function \(P:D \rightarrow \Delta(S)\)
    - \(\Delta(S)\) is the set of distributions over \(S\)
  - Utility function \(U\) over \(S\)

• A solution is any \(d^*\) in \(D\) such that \(EU(d^*) \geq EU(d)\) for all \(d\) in \(D\)
• This viewpoint accounts for
  - Uncertainty in action outcomes
  - Uncertainty in state of knowledge
  - Any combination of the two
Notes: Expected Utility

• Why Maximum Expected Utility?

• Where do these utilities come from?
  - Preference elicitation
• Utility functions need not be unique
  - If you multiply \( U \) by a positive constant, all decisions have the same relative utility
  - If you add a constant to \( U \), then the same thing is true
• \( U \) is unique up to a positive affine transformation

If \( d^* = \text{argmax}_d \ Pr(d)U(d) \) then
\[
d^* = \text{argmax}_d Pr(d)[aU(d) + b]
\]
a > 0
What are the Complications?

• Outcome space can be large
  - State space can be huge
  - Do not want to spell out distributions explicitly
  - **Solution**: Use Bayes Nets (or related Influence diagrams)

• Decision space is large
  - Usually decisions are not one-shot
    - Sequential choice
    - If we treat each plan as a distinct decision, then the space is too large to handle directly
  - **Solution**: Use dynamic programming to construct optimal plans
What are the Complications?

• Outcome space can be large
  • State space can be huge
  • Do not want to spell out distributions explicitly
  • **Solution**: Use Bayes Nets (or related Influence diagrams)

• Decision space is large
  • Usually decisions are not one-shot
    • Sequential choice
    • If we treat each plan as a distinct decision, then the space is too large to handle directly
  • **Solution**: Use dynamic programming to construct optimal plans
Simple Example

• Two actions: a, b
  - That is, either [a,a], [a,b], [b,a], [b,b]
• We can execute two actions in sequence
• Actions are stochastic: action a induces distribution $P_a(s_i|s_j)$ over states
  - $P_a(s_2|s_1)=0.9$ means that the prob. of moving to state $s_2$ when taking action a in state $s_1$ is 0.9
  - Similar distribution for action b
• How good is a particular plan?
Distributions for Action Sequences

![Diagram showing distributions for action sequences with nodes s1, s2, s3, s12, and s13, and actions a and b with associated probabilities.](image-url)
How Good is a Sequence?

• We associate utilities with the **final outcome**
  - How good is it to end up at $s_4$, $s_5$, $s_6$, ...

• Now we have:
  - $EU(aa) = .45U(s_4) + .45U(s_5) + .02U(s_8) + .08(s_9)$
  - $EU(ab) = .54U(s_6) + .36U(s_7) + .07U(s_{10}) + .03U(s_{11})$
  - etc
Utilities for Action Sequences

Looks a lot like a game tree, but with chance nodes instead of min nodes. (We average instead of minimizing)
Why Sequences Might Be Bad

• Suppose we do \( a \) first; we could reach \( s_2 \) or \( s_3 \)
  - At \( s_2 \), assume: \( EU(a) = 0.5U(s_4) + 0.5U(s_5) > EU(b) = 0.6U(s_6) + 0.4U(s_7) \)
  - At \( s_3 \), assume: \( EU(a) = 0.2U(s_8) + 0.8U(s_9) < EU(b) = 0.7U(s_{10}) + 0.3U(s_{11}) \)

• After doing \( a \) first, we want to do \( a \) next if we reach \( s_2 \), but we want to be \( b \) second if we reach \( s_3 \)
Policies

• We want to consider **policies**, not sequences of actions (plans)

• We have 8 policies for the decision tree:

  - [a; if s2 a, if s3 a]  [b; if s12 a, if s13 a]
  - [a; if s2 a, if s3 b]  [b; if s12 a, if s13 b]
  - [a; if s2 b, is s3 a]  [b; if s12 b, if s13 a]
  - [a; if s2 b, if s3 b]  [b; if s12 b, if s13 b]

• We have 4 plans

  - [a;a], [a;b], [b;a], [b;b]

  - **Note**: each plans corresponds to a policy so we can only **gain** by allowing the decision maker to use policies
Evaluating Policies

• Number of plans (sequences) of length k
  - Exponential in k: \( |A|^k \) if A is the action set

• Number of policies is much larger
  - If A is the action set and O is the outcome set, then we have \( (|A||O|)^k \) policies

• Fortunately, dynamic programming can be used
  - Suppose \( EU(a) > EU(b) \) at s2
  - Never consider a policy that does anything else at s2

• How to do this?
  - Back values up the tree much like minimax search
Decision Trees

- Squares denote choice nodes (decision nodes)
- Circles denote chance nodes
  - Uncertainty regarding action effects
- Terminal nodes labelled with utilities
Evaluating Decision Trees

- Procedure is exactly like game trees except
  - “MIN” is “nature” who chooses outcomes at chance nodes with specified probability
    - Average instead of minimize

- Back values up the tree
  - U(t) defined for terminal nodes
  - U(n)=avg {U(c):c a child of n} if n is chance node
  - U(n)=max{U(c:c is child of n)} if n is a choice node
Evaluating a Decision Tree

![Decision Tree Diagram]
Decision Tree Policies

• Note that we don’t just compute values, but policies for the tree
• A **policy** assigns a decision to each choice node in the tree
• Some policies can’t be distinguished in terms of their expected values
  - Example: If a policy chooses a at s1, the choice at s4 does not matter because it won’t be reached
  - Two policies are **implementationally indistinguishable** if they disagree only on unreachable nodes
Computational Issues

• Savings compared to explicit policy evaluation is substantial

• Let \( n = |A| \) and \( m = |O| \)
  - Evaluate only \( O((nm)^d) \) nodes in tree of depth \( d \)
    - Total computational cost is thus \( O((nm)^d) \)
  - Note that there are also \( (nm)^d \) policies
    - Evaluating a single policy requires \( O(m^d) \)
    - Total computation for explicitly evaluating each policy would be \( O(n^d m^{2d}) \)
Computational Issues

Tree size: Grows exponentially with depth
- Possible solutions: Bounded lookahead, heuristic search procedures

Full Observability: We must know the initial state and outcome of each action
- Possible solutions: Handcrafted decision trees, more general policies based on observations
Other Issues

**Specification:** Suppose each state is an assignment of values to variables

- Representing action probability distributions is complex
  - Large branching factor

• Possible solutions:
  - Bayes Net representations
  - Solve problems using decision networks

We will discuss these later in the semester
Key Assumption: Observability

**Full observability**: We must know the initial state and outcome of each action

- To implement a policy we must be able to resolve the uncertainty of any chance node that is followed by a decision node
  - e.g. After doing a at s1, we must know which of the outcomes (s2 or s3) was realized so that we know what action to take next

- Note: We don’t need to resolve the uncertainty at a chance node if no decision follows it
Partial Observability

- If we push (unobservable) uncertainty to the “end of the tree” then we can evaluate the tree
  - often used in handcrafted decision trees

Here we push uncertainty re: disease to end of tree. All chance outcomes preceding decision are fully observable.
Large State Spaces (Variables)

• To represent outcomes of actions or decisions, we need to specify distributions
  - $P(s|d)$: probability of outcome $s$ given decision $d$
  - $P(s|a,s')$: probability of state $s$ given action $a$ was taken in state $s'$
• Note that the state space is exponential in the number of variables
  - Spelling out distributions explicitly is intractable
• Bayes Nets can be used to represent actions
  - Joint distribution over variables, conditioned on action/decision and previous state
Summary

• Basic properties of preferences
• Relationship between preferences and utilities
• Principle of Maximum Expected Utility
• Decision Trees