Markov Decision Processes

CSE 473 May 28, 2004

• *AI textbook: Sections 17.2-17.4* - Russel and Norvig
• *Decision-Theoretic Planning: Structural Assumptions and Computational Leverage* - Boutilier, Dean, Hanks
Planning

Environment

Static vs. Dynamic

Instantaneous vs. Deterministic

Fully Observable vs. Partially Observable

Perfect vs. Noisy

Perceps

What action next?

Actions

Instantaneous vs. Durative

Deterministic vs. Stochastic

Full vs. Partial satisfaction
Planning under uncertainty

Environment

Static?

Perfect
Partially Observable

Instantaneous
Stochastic

What action next?

Full
Uncertainty

• Disjunctive
  Next state could be one of a set of states.

• Probabilistic
  Next state is a probability distribution over the set of states.

Which is a more general model?
Probabilistic uncertainty

- Probability of new state could be dependent on the history of states visited.
  \[ \Pr(s^t|s^{t-1}, s^{t-2}, \ldots, s^0) \]

- First Order Markovian Assumption
  \[ \Pr(s^t|s^{t-1}, s^{t-2}, \ldots, s^0) = \Pr(s^t|s^{t-1}) \]

- Stationary Model assumption
  \[ \Pr(s^t|s^{t-1}) = \Pr(s^k|s^{k-1}) \text{ for all } k. \]
A general stochastic process (a), a Markov chain (b), and a stationary Markov chain (c).

What is the cost of representing a stationary probability distribution?
A Factored domain

• Variables:
  has_user_coffee (huc), has_robot_coffee (hrc), robot_is_wet (w), has_robot_umbrella (u), raining (r), robot_in_office (o)

• Actions:
  buy_coffee, deliver_coffee, get_umbrella, move

What is the number of states?
Can we succinctly represent transition probabilities in this case?
Dynamic Bayesian Nets

Total values required to represent transition probability table = 36

Total values required for a complete state probability table?
Exogenous events

• Example: Rain
• Do not occur as a result of our actions...
• One way to incorporate these in our model:
  
  Add them explicitly as a result of each action with some probability.
  
  Ex. After every action there is a low probability of “raining” changing its value.
Observability

- Full Observability
- Partial Observability
- No Observability
Reward/cost

- Each action has an associated cost.
- Agent may accrue rewards at different stages. A reward may depend on
  - The current state
  - The (current state, action) pair
  - The (current state, action, next state) triplet
- Additivity assumption: Costs and rewards are additive.
- Reward accumulated = $R(s^0)+R(s^1)+R(s^2)+...$
Horizon

- **Finite**: Plan till $t$ stages. Reward = $R(s^0) + R(s^1) + R(s^2) + \ldots + R(s^t)$
- **Infinite**: The agent never dies. The reward $R(s^0) + R(s^1) + R(s^2) + \ldots$ could be unbounded.

  - Discounted reward: $R(s^0) + \gamma R(s^1) + \gamma^2 R(s^2) + \ldots$
  - Average reward: $\lim_{n \to \infty} \frac{1}{n} \left[ \sum_i R(s^i) \right]$
Goal for an MDP

- Find a policy which:
  maximises expected discounted reward for
  an infinite horizon for a
  fully observable
  Markov decision process.

Why shouldn’t the planner find a plan??
What is a policy??
Optimal value of a state

- Define $V^*(s)$ {value of a state} as the maximum expected discounted reward achievable from this state.
- Value of state if we force it to do action "a" right now, but let it act optimally later:
  \[ Q^*(a,s) = R(s) + c(a) + \gamma \sum_{s' \in S} \Pr(s'|a,s)V^*(s') \]
- $V^*$ should satisfy the following equation:
  \[ V^*(s) = \max_{a \in A} \{Q^*(a,s)\} \]
  \[ = R(s) + \max_{a \in A} \{c(a) + \gamma \sum_{s' \in S} \Pr(s'|a,s)V^*(s')\} \]
Value iteration

• Assign an arbitrary assignment of values to each state (or use an admissible heuristic).

• Iterate over the set of states and in each iteration improve the value function as follows:
  \[ V_{t+1}(s) = R(s) + \max_{a \in A} \{ c(a) + \gamma \sum_{s' \in S} \Pr(s'|a, s) V_t(s') \} \] (Bellman Backup)

• Stop the iteration appropriately. \( V_t \) approaches \( V^* \) as \( t \) increases.
Bellman Backup

\[ V_{n+1}(s) \xrightarrow{Q_{n+1}(s,a)} Max \]

\[ a_1 \]
\[ a_2 \]
\[ a_3 \]

\[ V_n \]

\[ V_n \]
\[ V_n \]
\[ V_n \]
\[ V_n \]
\[ V_n \]
Stopping Condition

• \( \varepsilon \)-convergence: A value function is \( \varepsilon \)-optimal if the error (residue) at every state is less than \( \varepsilon \).

\[
\text{Residue}(s) = |V_{t+1}(s) - V_t(s)|
\]

Stop when \( \max_{s \in \mathcal{S}} R(s) < \varepsilon \)
Complexity of value iteration

• One iteration takes $O(|A||S|^2)$ time.
• Number of iterations required: 
  poly$(|S|,|A|,1/(1-\gamma))$
• Overall, the algorithm is polynomial in state space, and thus exponential in number of state variables.
Computation of optimal policy

- Given the value function $V^*(s)$, for each state, do Bellman backups and the action which maximises the inner product term is the optimal action.

- Optimal policy is stationary (time independent) - intuitive for infinite horizon case.
Policy evaluation

- Given a policy $\Pi: S \rightarrow A$, find value of each state using this policy.
- $V^\Pi(s) = R(s) + c(\Pi(s)) + \gamma[\sum_{s' \in S} Pr(s' | \Pi(s), s)V^\Pi(s')]$
- This is a system of linear equations involving $|S|$ variables.
Bellman’s principle of optimality

- A policy $\Pi$ is optimal if $V^\Pi(s) \geq V^{\Pi'}(s)$ for all policies $\Pi'$ and all states $s \in S$.
- Rather than finding the optimal value function, we can try and find the optimal policy directly, by doing a policy space search.
Policy iteration

• Start with any policy \((\Pi_0)\).

• Iterate

  Policy evaluation: For each state find \(V^{\Pi_i}(s)\).
  Policy improvement: For each state \(s\), find action \(a^*\) that maximises \(Q^{\Pi_i}(a,s)\).
  If \(Q^{\Pi_i}(a^*,s) > V^{\Pi_i}(s)\) let \(\Pi_{i+1}(s) = a^*\)
  else let \(\Pi_{i+1}(s) = \Pi_i(s)\)

• Stop when \(\Pi_{i+1} = \Pi_i\)

• Converges faster than value iteration but policy evaluation step is more expensive.
Modified Policy iteration

• Rather than evaluating the actual value of policy by solving system of equations, approximate it by using value iteration with fixed policy.
Other speedups

- Heuristics
- Aggregations
- Reachability Analysis
Going beyond full observability

• In execution phase, we are uncertain where we are,
• but we have some idea of where we can be.
• A belief state = some idea of where we are (represented as a set of/probability distribution over the states).
Mathematical modelling

• Search space: finite/infinite state/belief space.
  Belief state = some idea of where we are
• Initial state/belief.
• Actions
• Action transitions (state to state / belief to belief)
• Action costs
• Feedback: Zero/Partial/Total
Full Observability

• Modelled as MDPs. (also called fully observable MDPs)
• Output: Policy (State $\rightarrow$ Action)
• Bellman Equation

$$V^*(s) = \max_{a \in A(s)} \left[ c(a) + \sum_{s' \in S} V^*(s') P(s' | s, a) \right]$$
Partial Observability

- Modelled as POMDPs. (partially observable MDPs). Also called Probabilistic Contingent Planning.
- Belief = probabilistic distribution over states.
- What is the size of belief space?
- Output: Policy (Discretized Belief -> Action)
- Bellman Equation

\[ V^*(b) = \max_{a \in A(b)} [c(a) + \sum_{o \in O} P(b, a, o) V^*(b_{a^o})] \]
No observability

- Deterministic search in the belief space.
- Output ?