CSC242: Intro to AI

Lecture 19
Making Complex Decisions
Making Complex Decisions
Bayesian Networks

Random Variables

- \( Cavity \)
- \( Toothache \)
- \( Catch \)

\[
P(Cavity) \quad \text{“has direct influence on”} \quad P(Toothache \mid Cavity) \quad P(Catch \mid Cavity)
\]

\[
P(Toothache \mid Cavity)
\]

conditionally independent given parents

\[
P(X_i \mid \text{Parents}(X_i))
\]
Exact Inference in Bayesian Networks

• #P-Hard even for distribution described as a Bayesian Network
Approximate Inference in Bayesian Networks

- Sampling consistent with a distribution
- Rejection Sampling: rejects too much
- Likelihood Weighting: weights get too small
- Gibbs Sampling: MCMC algorithm (like local search)
- All generate consistent estimates (equal to exact probability in the large-sample limit)
\[ \text{Rain}_t = \text{R}_t \]

\[ \text{Umbrella}_t = \text{U}_t \]

\[ \text{X}_t : R_0, R_1, R_2, \ldots \]

\[ \text{E}_t : U_1, U_2, \ldots \]
\[ R_{t-1}, P(R_t) \]

\[
\begin{array}{|c|c|}
\hline
R_{t-1} & P(R_t) \\
\hline
t & 0.7 \\
f & 0.3 \\
\hline
\end{array}
\]

\[ R_t, P(U_t) \]

\[
\begin{array}{|c|c|}
\hline
R_t & P(U_t) \\
\hline
t & 0.9 \\
f & 0.2 \\
\hline
\end{array}
\]
Temporal Model

\[ P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1}^{t} P(X_i | X_{i-1}) P(E_i | X_i) \]

- Initial State Model
- Transition Model
- Sensor Model
Expected Utility

\[ EU(a \mid e) = \sum_{s'} P(\text{Result}(a) = s' \mid a, e) U(s') \]

- **Probabilistic inference**
- **Utility function**
- **Sum over possible outcomes**
Maximum Expected Utility

A rational agent should choose the action that maximizes the agent’s expected utility

\[ a_{MEU} = \arg\max_a EU(a \mid e) \]
Sequential Decision Problem

- Utility depends on a sequence of decisions
- Incorporate uncertainty, utility, sensing
- Include search and planning as special cases
$Actions(s) = \{ \text{Up, Down, Left, Right} \}$
[ Up, Up, Right, Right, Right ]
Nondeterministic Actions
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>START</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
[ \text{Up, Up, Right, Right, Right} ]
\]

\[
0.8^5 = 0.32768
\]
\[
\begin{array}{|c|c|c|}
\hline
& & +1 \\
\hline
& -1 & \\
\hline
\text{START} & & \\
\hline
\end{array}
\]

[ Up, Up, Right, Right, Right ]

\[0.1^4 \times 0.8 = 0.00008\]
Stochastic Transition Model

\[ P(s' \mid s, a) \]

\[ P(\text{Result}(a) = s' \mid a, e) \]
Markov Assumption

\[ P(s' \mid s, a) \]

\[ P(s' \mid s, s_{-1}, s_{-2}, \ldots, a) = P(s' \mid s, a) \]
Reward Function

- In each state $s$ the agent receives reward $R(s)$:
  - $R(s) = +1$ or $-1$ in terminal state
  - $R(s) = -0.04$ otherwise
Markov Decision Process

• Sequential decision problem
• Fully observable, stochastic environment
• Set of states $S$ with initial state $s_0$
• Markovian transition model: $P(s' \mid s, a)$
• Additive rewards: $R(s)$
[ Up, Up, Right, Right, Right ]
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
[ Up, Up, Right, Right, Right ]
```
Policy

• A policy $\pi$ specifies what the agent should do for any state the agent might reach: $\pi(s)$
Policy Quality

• A policy $\pi$ specifies what the agent should do for any state the agent might reach: $\pi(s)$

• Each time a policy is executed, it leads to a different history
Policy Quality

• A policy $\pi$ specifies what the agent should do for any state the agent might reach: $\pi(s)$

• Each time a policy is executed, it leads to a different history

• Quality of a policy is its expected utility
A policy $\pi$ specifies what the agent should do for any state the agent might reach: $\pi(s)$

Each time a policy is executed, it leads to a different history

Quality of a policy is its expected utility

Optimal policy $\pi^*$ maximizes expected utility
Optimal policies balance risk and reward
\[ R(s) < -1.6284 \]
-0.4278 < R(s) < -0.0850
\[-0.0221 < R(s) < 0\]
\( R(s) > 0 \)
\[ R(s) < -1.6284 \]

\[ -0.4278 < R(s) < -0.0850 \]

\[ -0.0221 < R(s) < 0 \]

\[ R(s) > 0 \]
Markov Decision Process

- Sequential decision problem
- Fully observable, stochastic environment
- Markovian transition model: $P(s' \mid s, a)$
- Additive rewards: $R(s)$
- Policy: $\pi(s) = \text{action to perform in state } s$
- Quality of a policy is its expected utility
- Optimal policy $\pi^*$ maximizes expected utility
Computing Optimal Policies

- Given MDP (transition & reward models)
- Compute optimal policy
- Inference!
<table>
<thead>
<tr>
<th></th>
<th>-0.04</th>
<th>-0.04</th>
<th>-0.04</th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.04</td>
<td></td>
<td>-0.04</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>-0.04</td>
<td>-0.04</td>
<td></td>
<td>-0.04</td>
<td></td>
</tr>
<tr>
<td>-0.04</td>
<td>-0.04</td>
<td>-0.04</td>
<td>-0.04</td>
<td></td>
</tr>
</tbody>
</table>

Rewards on states

Actions to reach states
\[ R(s_{4,3}) = +1 = U(s_{4,3}) \]
\( R(s_{3,3}) = -0.04 \)
\[ R(s_{3,3}) = -0.04 \]
\[ R(s_{4,3}) = +1 \]
\[ R(s_{3,3}) = -0.04 \]
\[ R(s_{4,3}) = +1 \]
\[ R(s_{3,3}) = -0.04 \]
\[ R(s_{4,3}) = +1 \]
\[ U(s_{3,3}) = R(s_{3,3}) + R(s_{3,4}) = 0.96 \]

Additive rewards
$R(s_{3,3}) = -0.04$

$R(s_{4,3}) = +1$

$U(s_{3,3}) = R(s_{3,3}) + \gamma R(s_{3,4})$

Discounted rewards
Stationary Preference

Given $S = [s_0, s_1, s_2, \ldots]$ and $S' = [s'_0, s'_1, s'_2, \ldots]$

If $s_0 = s'_0$ and $S \succeq S'$

Then $[s_1, s_2, \ldots] \succeq [s'_1, s'_2, \ldots]$
\[ R(s_{3,3}) = -0.04 \]
\[ R(s_{4,3}) = +1 \]
\[ U(s_{3,3}) = R(s_{3,3}) + \gamma R(s_{3,4}) \]
Discounted Reward

\[ U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots \]
Horizon

\[ U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots \]

- Finite horizon: Fixed time after which nothing matters (game is over)
- Optimal policy is nonstationary!
- Infinite horizon: no fixed deadline
Infinite Horizon

- Discounted rewards:

\[ U([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \]
Infinite Horizon

• Discounted rewards:

\[ U([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \]

• **Proper policy**: Guaranteed to reach a terminal state
Infinite Horizon

- Discounted rewards:
  \[
  U([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t)
  \]

- Proper policy: Guaranteed to reach a terminal state

- Compare policies in terms of average reward per time step
Utility of State Sequences

\[ U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots \]

\[ U([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \]
Utility of Policies

\[ U_\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(S_t) \right] \]

\[ S_0 = s \]
Utility of Policies

\[ U_\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(S_t) \right] \]

\[ U_\pi(s) = \sum_{[s_0, \ldots, s_t]} \sum_{t=0}^{\infty} \gamma^t R(S_t) P(S_t = s_t \mid s, \pi) \]
Optimal Policy

\[ U_\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(S_t) \right] \]

\[ \pi^*_s = \arg \max_\pi U^\pi(s) \]
\[ U^{\pi^*}(s) \]

\[ R(s) = -0.04, \quad \gamma = 1 \]

\[
\begin{array}{cccc}
0.812 & 0.868 & 0.918 & +\mid \\
0.762 & \text{ } & 0.660 & -\mid \\
0.705 & 0.655 & 0.611 & 0.388 \\
\end{array}
\]
Utility of Policies

\[ U_\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(S_t) \right] \]

\[ \pi_s^* = \arg\max_\pi U^\pi(s) \]
Computing Optimal Policies

• The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming the agent chooses the optimal action.
\[ U(s) = R(s) + \gamma EU(a^* | s) \]
Bellman Equation

\[ U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s') \]

Richard Bellman
(1920-1984)
Bellman Equation

\[ U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s') \]

- \( n \) states: \( n \) Bellman equations, one for each state, each with \( n \) unknowns
- Not linear: \( \max \)
- Have to solve iteratively
Value Iteration

• Easy to understand (AIMA 17.2.1-17.2.2)
• Computes utility function for optimal policy
• From which you can extract the policy (i.e., which action is best)
Value Iteration

\[ U_0(s) \leftarrow ? \text{ for all } s \]

\[ \text{do } \{ \]
\[ \text{for each state } s \{ \]
\[ U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U_i(s') \]
\[ \}
\[ \}
\[ \text{until (!changed_);} \]
Policy Iteration

- Searches space of policies, rather than refining values of utilities
Policy Iteration

\[ \pi(s) \leftarrow \text{? for all } s \]
do { 

\[ U \leftarrow \text{Policy-Evaluation}(\pi, U, mdp) \]

for each state \( s \) { 

let \( a \leftarrow \text{argmax} \sum_{s'} P(s' | s, a)U(s) \); \( U_a(s) \)

if \( U_a(s) > \sum_{s'} P(s' | s, \pi(s))U(s') \)

then \( \pi(s) \leftarrow a \)

}

} until (!changed_);
Policy Iteration

• Searches space of policies, rather than refining values of utilities

• Simpler than solving Bellman equations (ala value iteration)

• \( n \) linear equations in \( n \) unknowns: \( O(n^3) \)
Policy Iteration

- Searches space of policies, rather than refining values of utilities
- Simpler than solving Bellman equations (ala value iteration)
- $n$ linear equations in $n$ unknowns: $O(n^3)$
- Can be made into heuristic search process (asynchronous policy iteration)
MDP Summary

- Fully observable, stochastic environment
- Markovian transition model: $P(s' \mid s, a)$
- Additive or discounted rewards: $R(s)$
- Policy: $\pi(s) = \text{action to perform in state } s$
- Quality of a policy is its expected utility
- Optimal policy $\pi^*$ maximizes expected utility
- Value Iteration & Policy Iteration
Further Reading

- Partially-observable environments
- POMDPs
- Multi-agent systems
- AIMA 17.5: Game Theory
- AIMA 17.6: Design of MAS
Reinforcement Learning

- Computing optimal policy requires transition model and reward model
- Reinforcement learning tries to learn the reward model while learning the policy
- Harder!
For Next Time
18.0-18.4