Decision Theory and Markov Decision Processes (MDPs)

Ron Parr
CPS 271

The Winding Path to RL

- Decision Theory
- Markov Decision Processes
- Reinforcement Learning
- Descriptive theory of optimal behavior
- Mathematical/Algorithmic realization of Decision Theory
- Application of learning techniques to challenges of MDPs with numerous or unknown parameters
Covered in First Lecture

- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

Decision Theory

What does it mean to make an optimal decision?

- Asked by economists to study consumer behavior
- Asked by MBAs to maximize profit
- Asked by leaders to allocate resources
- Asked in OR to maximize efficiency of operations
- Asked in AI to model intelligence

- Asked (sort of) by any intelligent person every day
Utility Functions

- A utility function is a mapping from world states to real numbers
- Also called a value function
- Rational or optimal behavior is typically viewed as maximizing expected utility:

$$
\max_a \sum_s P(s \mid a)U(s)
$$

a = actions, s = states

Are Utility Functions Natural?

- Some have argued that people don’t really have utility functions
  - What is the utility of the current state?
  - What was your utility at 8:00pm last night?
  - Utility elicitation is difficult problem

- It’s easy to communicate preferences
- Given a plausible set of assumptions about preferences, must exist consistent utility function

(More precise statement of this is a theorem.)
Swept under the today

- Utility of money (assumed 1:1)
- How to determine costs/utilities
- How to determine probabilities

Playing a Game Show

- Assume series of questions
  - Increasing difficulty
  - Increasing payoff
- Choice:
  - Accept accumulated earnings and quit
  - Continue and risk losing everything
- “Who wants to be a millionaire?”
State Representation (simplified game)

Start
$100

1 correct
$1,000

2 correct
$10,000

3 correct
$50,000

$61,100

Making Optimal Decisions

- Work backwards from future to present

- Consider $50,000 question
  - Suppose P(correct) = 1/10
  - V(stop)=$11,100
  - V(continue) = 0.9*0 + 0.1*$61.1K = $6.11K

- Optimal decision stops
Working Recursively

\[ V = \frac{9}{10} \times 0 + \frac{3}{4} \times 0 + \frac{1}{2} \times 0 + \frac{1}{10} \times 100 \]

\[ = \frac{9}{10} \times 100 = 90 \]

\[ V = \frac{9}{10} \times 1,100 = 990 \]

\[ V = \frac{3}{4} \times 11,100 = 8325 \]

\[ V = \frac{1}{2} \times 11,100 = 5550 \]

Decision Theory Review

- Provides theory of optimal decisions
- Principle of maximizing utility
- Easy for small, tree structured spaces with
  - Known utilities
  - Known probabilities
Covered in Today

- Decision Theory
- MDPs

- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

Dealing with Loops

Suppose you can pay $1000 (from any losing state) to play again

```
$\begin{array}{c}
\text{9/10} \\
\text{3/4} \\
\text{1/2} \\
\text{1/10}
\end{array}
\begin{array}{c}
\text{$0} \\
\text{$0} \\
\text{$0} \\
\text{$0}
\end{array}
\begin{array}{c}
\text{9/10} \\
\text{3/4} \\
\text{1/2} \\
\text{1/10}
\end{array}
\begin{array}{c}
\text{$0} \\
\text{$0} \\
\text{$0} \\
\text{$0}
\end{array}
```

\(-1000\)$

\$100 \quad \$1,100 \quad \$11,100\)
From Policies to Linear Systems

• Suppose we always pay until we win.
• What is value of following this policy?

\[
\begin{align*}
V(s_0) &= 0.10(-1000 + V(s_0)) + 0.90V(s_1) \\
V(s_1) &= 0.25(-1000 + V(s_0)) + 0.75V(s_2) \\
V(s_2) &= 0.50(-1000 + V(s_0)) + 0.50V(s_3) \\
V(s_3) &= 0.90(-1000 + V(s_0)) + 0.10(61100)
\end{align*}
\]

And the solution is...

\[
\begin{align*}
V &= $3,749 & V &= $4,166 & V &= $5,555 & V &= $11.11K \\
V &= $32.47K & V &= $32.58K & V &= $32.95K & V &= $34.43K
\end{align*}
\]

Is this optimal?
How do we find the optimal policy?
The MDP Framework

• State space: S
• Action space: A
• Transition function: P
• Reward function: R
• Discount factor: $\gamma$
• Policy: $\pi(s) \rightarrow a$

Objective: *Maximize expected, discounted return* (decision theoretic optimal behavior)

Applications of MDPs

• AI/Computer Science
  – Robotic control
    (Koenig & Simmons, Thrun et al., Kaelbling et al.)
  – Air Campaign Planning (Meuleau et al.)
  – Elevator Control (Barto & Crites)
  – Computation Scheduling (Zilberstein et al.)
  – Control and Automation (Moore et al.)
  – Spoken dialogue management (Singh et al.)
  – Cellular channel allocation (Singh & Bertsekas)
Applications of MDPs

• Economics/Operations Research
  – Fleet maintenance (Howard, Rust)
  – Road maintenance (Golabi et al.)
  – Packet Retransmission (Feinberg et al.)
  – Nuclear plant management (Rothwell & Rust)

• EE/Control
  – Missile defense (Bertsekas et al.)
  – Inventory management (Van Roy et al.)
  – Football play selection (Patek & Bertsekas)

• Agriculture
  – Herd management (Kristensen, Toft)
The Markov Assumption

• Let $S_t$ be a random variable for the state at time $t$

• $P(S_t | A_{t-1}, S_{t-1}, \ldots, A_0, S_0) = P(S_t | A_{t-1}, S_{t-1})$

• Markov is a special kind of conditional independence

• Future is independent of past given current state

Understanding Discounting

• Mathematical motivation
  – Keeps values bounded
  – What if I promise you $0.01$ every day you visit me?

• Economic motivation
  – Discount comes from inflation
  – Promise of $1.00$ in future is worth $0.99$ today

• Probability of dying
  – Suppose $\epsilon$ probability of dying at each decision interval
  – Transition w/ prob $\epsilon$ to state with value 0
  – Equivalent to $1 - \epsilon$ discount factor
Discounting in Practice

- Often chosen unrealistically low
  - Faster convergence
  - Slightly myopic policies

- Can reformulate most algs for avg reward
  - Mathematically uglier
  - Somewhat slower run time

Covered Today

- Decision Theory
- MDPs

- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming
Value Determination

Determine the value of each state under policy $\pi$

$$V(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V(s')$$

Bellman Equation

Matrix Form

$$P = \begin{pmatrix}
P(s_1 \mid s_1, \pi(s_1)) & P(s_2 \mid s_1, \pi(s_1)) & P(s_3 \mid s_1, \pi(s_1)) \\
P(s_1 \mid s_2, \pi(s_2)) & P(s_2 \mid s_2, \pi(s_2)) & P(s_3 \mid s_2, \pi(s_2)) \\
P(s_1 \mid s_3, \pi(s_3)) & P(s_2 \mid s_3, \pi(s_3)) & P(s_3 \mid s_3, \pi(s_3))
\end{pmatrix}$$

$$V = \gamma P_{\pi} V + R$$

How do we solve this system?
Solving for Values

\[ V = \gamma P_\pi V + R \]

For moderate numbers of states we can solve this system exactly:

\[ V = (I - \gamma P_\pi)^{-1} R \]

Guaranteed invertible because \( \gamma P_\pi \) has spectral radius <1

Iteratively Solving for Values

\[ V = \gamma P_\pi V + R \]

For larger numbers of states we can solve this system indirectly:

\[ V^{i+1} = \gamma P_\pi V^i + R \]

Guaranteed convergent because \( \gamma P_\pi \) has spectral radius <1
Establishing Convergence

- Eigenvalue analysis

- Monotonicity
  - Assume all values start pessimistic
  - One value must always increase
  - Can never overestimate

- Contraction analysis...

Contraction Analysis

- Define maximum norm
  \[ \|V\|_\infty = \max_i V_i \]

- Consider V1 and V2
  \[ \|V_1 - V_2\|_\infty = \varepsilon \]

- WLOG say
  \[ V_1 \leq V_2 + \vec{\varepsilon} \]  (Vector of all ε’s)
Contraction Analysis Contd.

- At next iteration for $V_2$:
  $$V_2' = R + \gamma PV_2$$

- For $V_1$
  $$V_1 = R + \gamma P(V_1) \leq R + \gamma P(V_1 + \tilde{\epsilon}) = R + \gamma PV_1 + \gamma P\tilde{\epsilon} = R + \gamma PV_1 + \gamma \tilde{\epsilon}$$

- Conclude:
  $$\|V_2' - V_1'\|_\infty \leq \gamma \varepsilon$$

Importance of Contraction

- Any two value functions get closer

- True value function $V^*$ is a fixed point

- Max norm distance from $V^*$ decreases dramatically quickly with iterations
  $$\|V^{(0)} - V^*\|_\infty = \varepsilon \rightarrow \|V^{(n)} - V^*\|_\infty \leq \gamma^n \varepsilon$$

NB: (Superscripts) indicate iterations here
Iterative Policy Evaluation

-1000 9/10 3/4 1/2 1/10

<table>
<thead>
<tr>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>-100.00</td>
<td>-250.00</td>
<td>-500.00</td>
<td>5210.00</td>
</tr>
<tr>
<td>-335.00</td>
<td>-650.00</td>
<td>2055.00</td>
<td>4908.00</td>
</tr>
<tr>
<td>-718.50</td>
<td>1207.50</td>
<td>1892.50</td>
<td>9908.50</td>
</tr>
<tr>
<td>914.90</td>
<td>989.75</td>
<td>1595.00</td>
<td>4563.35</td>
</tr>
<tr>
<td>882.26</td>
<td>1174.97</td>
<td>2239.12</td>
<td>6033.41</td>
</tr>
</tbody>
</table>

Iterations

Iterations Continued

<table>
<thead>
<tr>
<th>iteration</th>
<th>V(S_0)</th>
<th>V(S_1)</th>
<th>V(S_2)</th>
<th>V(S_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>-100.0</td>
<td>-250.0</td>
<td>-500.0</td>
<td>5210.0</td>
</tr>
<tr>
<td>2</td>
<td>-335.0</td>
<td>-650.0</td>
<td>2055.0</td>
<td>4908.0</td>
</tr>
<tr>
<td>3</td>
<td>-718.5</td>
<td>1207.5</td>
<td>1892.5</td>
<td>9908.5</td>
</tr>
<tr>
<td>4</td>
<td>914.9</td>
<td>989.8</td>
<td>1595.0</td>
<td>4563.4</td>
</tr>
<tr>
<td>5</td>
<td>882.3</td>
<td>1175.0</td>
<td>2239.1</td>
<td>6033.4</td>
</tr>
<tr>
<td>10</td>
<td>2604.5</td>
<td>3166.7</td>
<td>4158.8</td>
<td>7241.8</td>
</tr>
<tr>
<td>20</td>
<td>5994.8</td>
<td>6454.5</td>
<td>7356.0</td>
<td>10.32K</td>
</tr>
<tr>
<td>200</td>
<td>29.73K</td>
<td>29.25K</td>
<td>29.57K</td>
<td>31.61K</td>
</tr>
<tr>
<td>2000</td>
<td>32.47K</td>
<td>32.58K</td>
<td>32.95K</td>
<td>34.43K</td>
</tr>
</tbody>
</table>

Note: Slow convergence because $\gamma=1$
Covered Today

- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

Finding Good Policies

Suppose an expert told you the “value” of each state:

\[
V(S1) = 10 \quad \text{V}(S2) = 5
\]
Improving Policies

• How do we get the optimal policy?
• Take the optimal action in every state
• Fixed point equation with choices:

\[ V^* (s) = \max_a \sum_{s'} R(s,a) + \gamma P(s'|s,a)V^* (s') \]

Decision theoretic optimal choice given \( V^* \)

Value Iteration

We can’t solve the system directly with a max in the equation
Can we solve it by iteration?

\[ V^{i+1} (s) = \max_a \sum_{s'} R(s,a) + \gamma P(s'|s,a)V^i (s') \]

• Called value iteration or simply successive approximation
• Same as value determination, but we can change actions

• Convergence:
  • Can’t do eigenvalue analysis (not linear)
  • Still monotonic
  • Still a contraction in max norm (exercise)
  • Converges quickly
Optimality

- VI converges to optimal policy
- Why?
- Optimal policy is stationary
- Why?

Covered Today

- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming
Greedy Policy Construction

Pick action with highest expected future value:

$$\pi_v(s) = \arg \max_a R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s')$$

Expectation over next-state values

$$\pi_v = \text{greedy}(V)$$

Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal $V$

Guess $\pi_v = \pi_0$

$V_\pi = \text{value of acting on } \pi$

(solve linear system)

$\pi_v \leftarrow \text{greedy}(V_\pi)$

Repeat until policy doesn’t change

Guaranteed to find optimal policy

Usually takes very small number of iterations

Computing the value functions is the expensive part
Comparing VI and PI

- **VI**
  - Value changes at every step
  - Policy *may* change at every step
  - Many cheap iterations
- **PI**
  - Alternates policy/value updates
  - Solves for value of each policy *exactly*
  - Fewer, slower iterations (need to invert matrix)
- **Convergence**
  - Both are contractions in max norm
  - PI is *shockingly* fast in practice (why?)

---

**Linear Programming**

\[ V(s) = R(s, a) + \gamma \max_a \sum_{s'} P(s'|s, a)V(s') \]

**Issue:** Turn the non-linear max into a collection of linear constraints

\[ \forall s, a : V(s) \geq R(s, a) + \gamma \sum_{s'} P(s'|s, a)V(s') \]

**MINIMIZE:** \[ \sum_s V(s) \]

Optimal action has tight constraints

Weakly polynomial; slower than PI in practice.
MDP Difficulties → RL

- MDP operate at the level of *states*
  - States = atomic events
  - We usually have exponentially (infinitely) many of these
- We assume P and R are known

- Machine learning to the rescue!
  - Infer P and R (implicitly or explicitly from data)
  - Generalize from small number of states/policies