Decision Theory and Markov Decision Processes (MDPs)

Ron Parr
CPS 271

DUKE COMPUTER SCIENCE

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.)

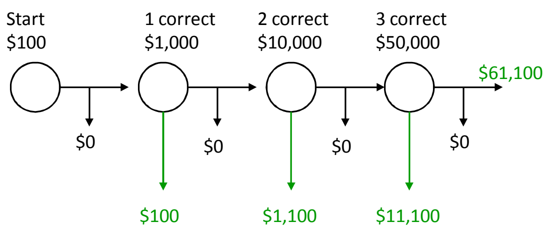
Swept under the today

- 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?"

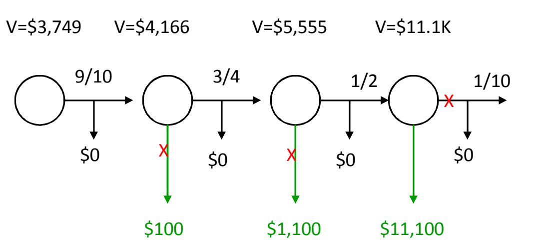
State Representation (simplified game)



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 continues

Working Recursively



Decision Theory Review

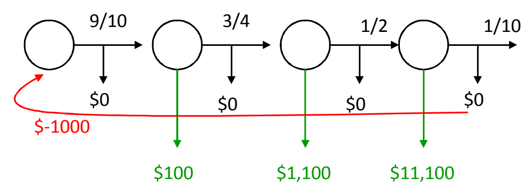
- Provides theory of optimal decisions
- Principle of maximizing utility
- Easy for small, tree structured spaces with
  - Known utilities
  - Known probabilities

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



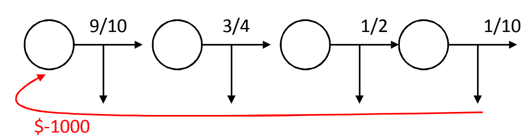
From Policies to Linear Systems

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

$$
\begin{aligned}
V(s_0) &= \underbrace{0.10(-1000+V(s_0))}_{\text{Return to Start}} + \underbrace{0.90V(s_1)}_{\text{Continue}} \\
V(s_1) &= 0.25(-1000+V(s_0)) + 0.75V(s_2) \\
V(s_2) &= 0.50(-1000+V(s_0)) + 0.50V(s_3) \\
V(s_3) &= 0.90(-1000+V(s_0)) + 0.10(61100)
\end{aligned}
$$

And the solution is…

V=$3,749	V=$4,166	V=$5,555	V=$11.11K	w/o cheat
V=$32.47K	V=$32.58K	V=$32.95K	V=$34.43K	



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: γ
- Policy: $\pi(s) \rightarrow a$

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

Applications of MDPs

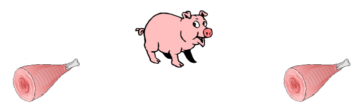
- AI/Computer Science
  - Robotic control
    (Koenig & Simmons, Thrun et al., Kaelbling et al.)
  - Air Campaign Planning (Meuleau et al.)
  - Elevator Control (Barto & Crites)
  - Computation Scheduling (Zilberstein et al.)
  - Control and Automation (Moore et al.)
  - Spoken dialogue management (Singh 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)

Applications of MDPs

- 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
- $P(S_t|A_{t-1}S_{t-1},\ldots,A_0S_0) = P(S_t|A_{t-1}S_{t-1})$
- Markov is 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 e probability of dying at each decision interval
  - Transition w/prob ε to state with value 0
  - Equivalent to 1- ε 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

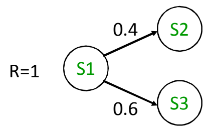
- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

Value Determination

Determine the value of each state under policy p

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

Bellman Equation



$$V(s_1) = 1 + \gamma(0.4V(s_2) + 0.6V(s_3))$$

Matrix Form

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

$$\mathbf{V} = \gamma \mathbf{P}_\pi \mathbf{V} + \mathbf{R}$$

How do we solve this system?

Solving for Values

$$\mathbf{V} = \gamma \mathbf{P}_\pi \mathbf{V} + \mathbf{R}$$

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

$$\mathbf{V} = \underbrace{(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}}\mathbf{R}$$

Guaranteed invertible because $\gamma \mathbf{P}_\pi$ has spectral radius <1

Iteratively Solving for Values

$$\mathbf{V} = \gamma \mathbf{P}_\pi \mathbf{V} + \mathbf{R}$$

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

$$\mathbf{V}^{i+1} = \gamma \mathbf{P}_\pi \mathbf{V}^{i} + \mathbf{R}$$

Guaranteed convergent because $\gamma \mathbf{P}_\pi$ 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

$$V_1 \le V_2 + \vec{\varepsilon} \qquad \text{(Vector of all ε's)}$$

Contraction Analysis Contd.

- At next iteration for V_2:

$$V_{2'} = R + \gamma P V_2$$

- For V_1

$$V_{1'} = R + \gamma P(V_1) \le R + \underbrace{\gamma P(V_2 + \vec{\varepsilon})}_{\text{Distribute}} = R + \gamma P V_2 + \gamma P \vec{\varepsilon} = R + \gamma P V_2 + \gamma \vec{\varepsilon}$$

- Conclude:

$$\left\| V_{2'} - V_{1'} \right\|_\infty \le \gamma \varepsilon$$

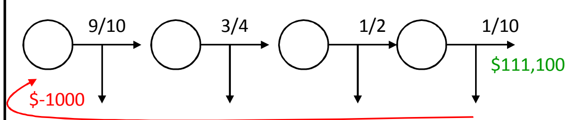
Importance of Contraction

- Any two value functions get closer
- True value function V* is a fixed point
- Max norm distance from V* decreases *exponentially* quickly with iterations

$$\left\| V^{(0)} - V^* \right\|_\infty = \varepsilon \rightarrow \left\| V^{(n)} - V^* \right\|_\infty \le \gamma^n \varepsilon$$

NB: (Superscripts) indicate iterations here

Iterative Policy Evaluation



0.00	0.00	0.00	0.00
-100.00	-250.00	-500.00	5210.00
-335.00	-650.00	2055.00	4908.00
-718.50	1207.50	1892.50	9908.50
914.90	989.75	1595.00	4563.35
882.26	1174.97	2239.12	6033.41

Iterations

Iterations Continued

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

Note: Slow convergence because γ=1

Covered Today

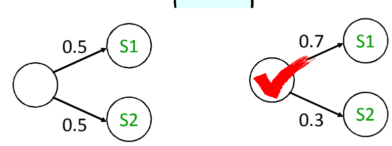
- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

Finding Good Policies

Suppose an expert told you the "value" of each state:

V(S1) = 10

V(S2) = 5



Action 1

Action 2

Improving Policies

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

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

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?

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

- 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 exponentially 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:

$$\pi(s) = \arg\max_a R(s,a) + \gamma \underbrace{\sum_{s'} P(s' \mid s,a) V(s')}_{\text{Expectation over next-state values}}$$

$$\pi = \mathrm{greedy}(V)$$

Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal V

Guess $\pi = \pi_0$
V_π = value of acting on π
(solve linear system)
$\pi \leftarrow \mathrm{greedy}(V_\pi)$

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 udpates
  - 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 in 1 Slide

- Minimize: $\mathbf{c}^T\mathbf{x}$
- Subject to: $\mathbf{Ax} \geq \mathbf{b}$
- Can be solved in weakly polynomial time
- Arguably most common and important optimization technique in history

Linear Programming

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

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

$$\forall s,a : V(s) \geq \underbrace{R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s')}_{\text{Optimal action has tight constraints}}$$

MINIMIZE: $\sum_s V(s)$

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 assumes 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