Markov Decision Processes (MDPs)

Ron Parr
CPS 570

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 Today

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

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

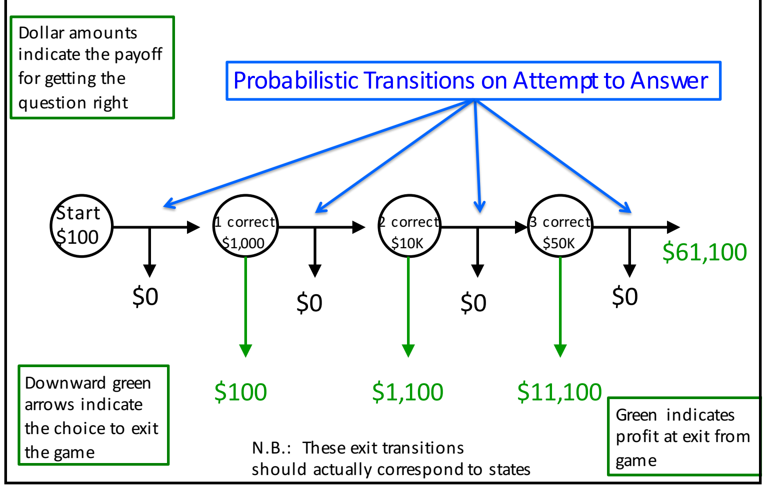
Swept under the rug today

- 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?"

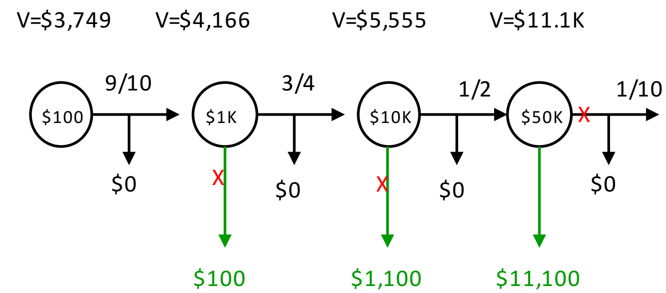
State Representation



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 Backwards



Red X indicates bad choice

Decision Theory Review

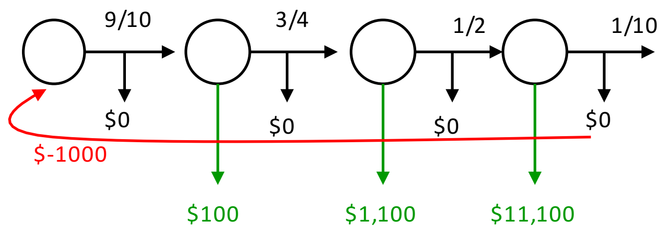
- Provides theory of optimal decisions
- Principle of maximizing utility
- Easy for small, tree structured spaces with
  - Known utilities
  - Known probabilities

Covered in Today

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

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?

$$
\begin{aligned}
V(s_0) &= 0.10(-1000 + V(s_0)) + 0.90V(s_1) \\
V(s_1) &= 0.25(-1000 + V(s_0)) + 0.75V(s_2) \\
V(s_2) &= 0.50(-1000 + V(s_0)) + 0.50V(s_3) \\
V(s_3) &= 0.90(-1000 + V(s_0)) + 0.10(61100)
\end{aligned}
$$

Return to Start Continue

And the solution is…

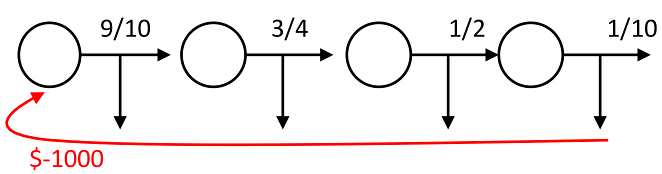
V=$3,749 → V=$32.47K

V=$4,166 → V=$32.58K

V=$5,555 → V=$32.95K

V=$11.11K → V=$34.43K

w/o cheat



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(s,a,s') or R(s,a) or R(s)
- Discount factor: γ
- Policy: $\pi(s) \rightarrow a$

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

Applications of MDPs

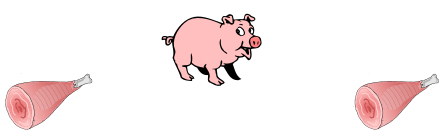
- AI/Computer Science
  - Robotic control
    (Koenig & Simmons, Thrun et al., Kaelbling et al.)
  - Air Campaign Planning (Meuleau et al.)
  - Elevator Control (Barto & Crites)
  - Computation Scheduling (Zilberstein et al.)
  - Control and Automation (Moore et al.)
  - Spoken dialogue management (Singh 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 ε 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 of the algorithms we'll see later
  - Leads to slightly myopic policies
- Can reformulate most algs. for avg. reward
  - Mathematically uglier
  - Somewhat slower run time

Covered Today

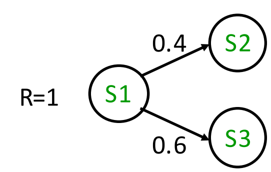
- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration

Value Determination

Determine the value of each state under policy π

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

Bellman Equation for a fixed policy π



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

Matrix Form

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

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

This is a generalization of the game show example from earlier

How do we solve this system efficient? Does it even have a solution?

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
  (don't worry if you don't know this)
- Monotonicity
  - Assume all values start pessimistic
  - One value must always increase
  - Can never overestimate
  - Easy to prove
- Contraction analysis…

Contraction Analysis

- Define maximum norm

$$\left\|V\right\|_\infty = \max_i V[i]$$

- Consider V1 and V2

$$\left\|V_i^a - V_i^b\right\|_\infty = \varepsilon$$

- WLOG say

$$V_i^a \le V_i^b + \vec{\varepsilon} \qquad \text{(Vector of all } \varepsilon\text{'s)}$$

Contraction Analysis Contd.

- At next iteration for V^b:

$$V_2^b = R + \gamma P V_1^b$$

- For V^a

$$V_2^a = R + \gamma P(V_1^a) \le R + \underbrace{\gamma P(V_1^b + \vec{\varepsilon})}_{\text{Distribute}} = R + \gamma P V_1^b + \gamma P\vec{\varepsilon} = R + \gamma P V_1^b + \gamma\vec{\varepsilon}$$

- Conclude:

$$\left\|V_2^a - V_2^b\right\|_\infty \le \gamma\varepsilon$$

Importance of Contraction

- Any two value functions get closer
- True value function V* is a fixed point (value doesn't change with iteration)
- Max norm distance from V* decreases *dramatically* quickly with iterations

$$\left\|V_0 - V^*\right\|_\infty = \varepsilon \rightarrow \left\|V_n - V^*\right\|_\infty \le \gamma^n \varepsilon$$

Covered Today

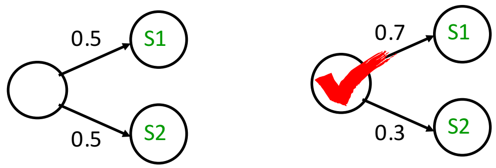
- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration

Finding Good Policies

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

V(S1) = 10

V(S2) = 5



Action 1

Action 2

Improving Policies

- How do we get the optimal policy?
- If we knew the values under the optimal policy, then just take the optimal action in every state
- How do we define these values?
- Fixed point equation with choices (Bellman equation):

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

Decision theoretic optimal choice given V*
If we know V*, picking the optimal action is easy
If we know the optimal actions, computing V* is easy
How do we compute both at the same time?

Value Iteration

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

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

- 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

Properties of Value Iteration

- VI converges to the optimal policy
  (implicit in the maximizing action at each state)
- Why? (Because we figure out V*)
- Optimal policy is stationary (i.e. Markovian – depends only on current state)
- Why? (Because we are summing utilities. Thought experiment: Suppose you think it's better to change actions the second time you visit a state. Why didn't you just take the best action the first time?)

Covered Today

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

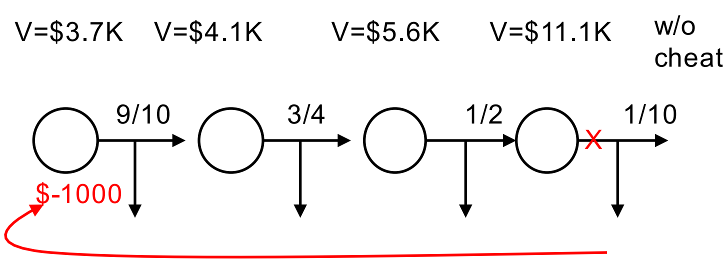
Greedy Policy Construction

Let's name the action that looks best WRT V:

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

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

Consider our first policy



Recall: We played until last state, then quit
Is this greedy with cheat option?

Value of paying to cheat in the first state is:
0.1(-1000 + 3.7K) + 0.9*(4.1K)=$3960
(much better than just giving up, which has value 0)

Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal V

Guess $\pi_v = \pi_0$
V_π = value of acting on π
(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

Computational Complexity

- VI and PI are both contraction mappings w/rate γ
  (we didn't prove this for PI in class)
- VI costs less per iteration
- For n states, a actions PI tends to take O(n) iterations in practice
  - Recent results indicate ~$O(n^2 a/1-\gamma)$ worst case
  - Interesting aside: Biggest insight into PI came ~50 years after the algorithm was introduced

MDP Difficulties → Reinforcement Learning

- MDP operate at the level of *states*
  - States = atomic events
  - We usually have exponentially (or 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

Advanced Topics

- Multiple agents
- Reinforcement Learning
- Partial observability