Markov Decision Processes
&
Reinforcement Learning

(Lecture 1)

Ron Parr
CPS 170

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 (quick review)
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

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

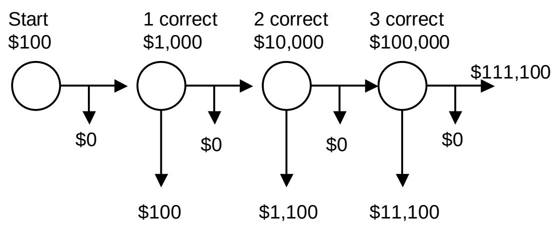
Swept under the rug today…

- 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?"

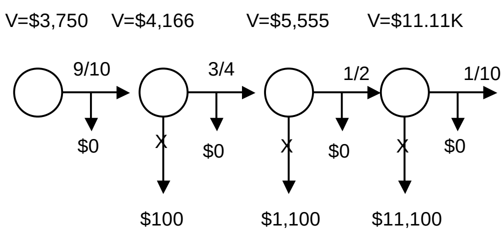
State Representation (simplified game)



Making Optimal Decisions

- Work *backwards* from future to present
- Consider $100,000 question
  - Suppose P(correct) = 1/10
  - V(stop)=$11,100
  - V(continue) = 0.9*$0 + 0.1*$111,100K = $11,110K
- Optimal decision CONTINUES through last step

Working Recursively



Decision Theory Summary

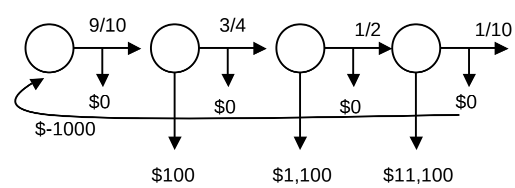
- Provides theory of optimal decisions
- Principle of maximizing utility
- Easy for small, tree structured spaces with
  - Known utilities
  - Known probabilities

Covered 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



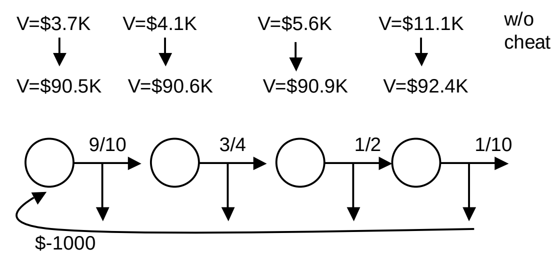
From Policies to Linear Systems

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

$$V(s_0) = 0.10(-1000 + V(s_0)) + 0.90V(s_1)$$
$$V(s_1) = 0.25(-1000 + V(s_0)) + 0.75V(s_2)$$
$$V(s_2) = 0.50(-1000 + V(s_0)) + 0.50V(s_3)$$
$$V(s_3) = 0.90(-1000 + V(s_0)) + 0.10(111100)$$

Return to Start — Continue

And the solution is…



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)

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

Covered Today

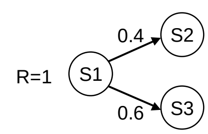
- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
    - Linear Programming

Value Determination

Determine the value of each state under policy π

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

Bellman Equation



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

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

$$V_1 \le V_2 + \vec{\varepsilon}$$

Contraction Analysis Contd.

- At next iteration for V2:

$$V^{2'} = R + \gamma P V^2$$

- For V1

$$V^{1'} = R + \gamma P(V^1) \le R + \gamma P(V^2 + \vec{\varepsilon}) = R + \gamma P V^2 + \gamma P \vec{\varepsilon} = R + \gamma P V^2 + \gamma \vec{\varepsilon}$$

Distribute

- Conclude:

$$\|V^{2'} - V^{1'}\|_\infty \le \gamma \varepsilon$$

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

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

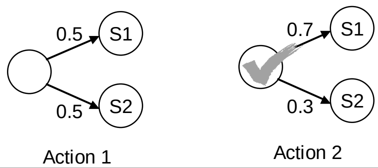
Covered Today

- 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



Improving Policies

- How do we get the optimal policy?
- Need to ensure that we take the optimal action in every state:

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

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 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

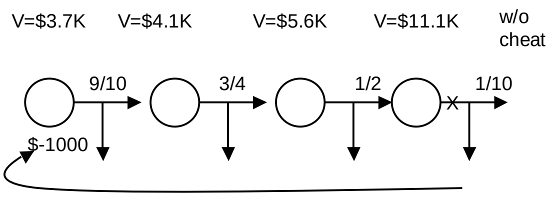
Greedy Policy Construction

Pick action with highest expected future value:

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

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

Consider our first policy



Recall: We played until last state, then quit

Is this greedy with cheat option?

Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal V

Guess V
π = greedy(V)
V = value of acting on π

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?)

Covered Today

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

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

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

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 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