CSE 473 Markov Decision Processes

Dan Weld

Many slides from Chris Bishop, Mausam, Dan Klein, Stuart Russell, Andrew Moore & Luke Zettlemoyer

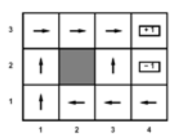
MDPs

Markov Decision Processes
- Planning Under Uncertainty
- Mathematical Framework
- Bellman Equations
- Value Iteration
- Real-Time Dynamic Programming
- Policy Iteration
- Reinforcement Learning

Andrey Markov (1856-1922)

Recap: Defining MDPs

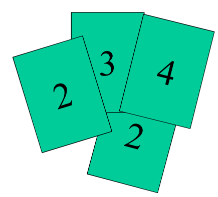
- Markov decision process:
  - States S
  - Start state s_0
  - Actions A
  - Transitions P(s'|s, a) aka T(s,a,s')
  - Rewards R(s,a,s') (and discount γ)



- Compute the optimal policy, π^*
  - π is a ***function*** that chooses an action for each state
  - We the policy which maximizes ***sum of discounted rewards***

High-Low as an MDP

- States:
  - 2, 3, 4, done
- Actions:
  - High, Low
- Model: T(s, a, s'):
  - P(s'=4 | 4, Low) = 1/4
  - P(s'=3 | 4, Low) = 1/4
  - P(s'=2 | 4, Low) = 1/2
  - P(s'=done | 4, Low) = 0
  - P(s'=4 | 4, High) = 1/4
  - P(s'=3 | 4, High) = 0
  - P(s'=2 | 4, High) = 0
  - P(s'=done | 4, High) = 3/4
  - …
- Rewards: R(s, a, s'):
  - Number shown on s' if s'<s $\wedge$ a="high" …
  - 0 otherwise
- Start: 3



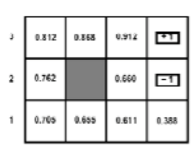
So, what's a policy?

π : {2,3,4,D} → {hi,lo}

Bellman Equations for MDPs

- <S, A, Pr, R, s_0, γ>
- Define **V*(s)** {optimal **value**} as the **maximum** expected **discounted reward** from this state.
- V* should satisfy the following equation:

$$V^*(s) = \max_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma V^*(s')\right]$$



Bellman Equations for MDPs

Q*(a, s)

$$V^*(s) = \max_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma V^*(s')\right]$$

$$V^*(s) = \max_a Q^*(s,a)$$

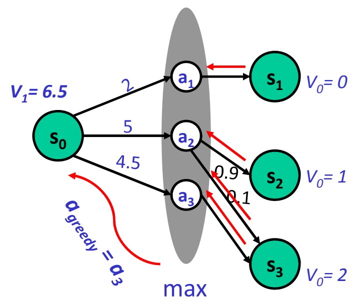
Bellman Backup (MDP)

- Given an estimate of V* function (say V_n)
- Backup V_n function at state s
  - calculate a new estimate (V_{n+1}) :

$$Q_{n+1}(s,a) = \sum_{s' \in \mathcal{S}} Pr(s'|s,a) \left[\mathsf{R}(s,a,s') + \gamma V_n(s') \right]$$
$$V_{n+1}(s) = \max_{a \in Ap(s)} [Q_{n+1}(s,a)]$$

- $Q_{n+1}(s,a)$: value/cost of the strategy:
  - execute action a in s, execute π_n subsequently
  - $\pi_n = \text{argmax}_{a \in Ap(s)} Q_n(s,a)$

Bellman Backup



$Q_1(s,a_1) = 2 + \gamma\, 0$
~ 2

$Q_1(s,a_2) = 5 + \gamma\, 0.9 \sim 1$
$+ \gamma\, 0.1 \sim 2$
~ 6.1

$Q_1(s,a_3) = 4.5 + \gamma\, 2$
~ 6.5

Value iteration [Bellman'57]

- assign an arbitrary assignment of V_0 to each state.
- repeat
  - for all states s
    - compute $V_{n+1}(s)$ by Bellman backup at s.
- until $\max_s |V_{n+1}(s) - V_n(s)| < \epsilon$

Iteration n+1

Residual(s)

- Theorem: will converge to unique optimal values
  - Basic idea: approximations get refined towards optimal values
  - ***Policy may converge*** long before values do

Example: Value Iteration

QuickTime™ and a GIF decompressor are needed to see this picture.

What about Policy?

- Which action should we chose from state s:
  - Given optimal values Q?

$$\arg\max_a Q^*(s,a)$$

  - Given optimal values V?

$$\arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

  - Lesson: actions are easier to select from Q's!

Policy Computation

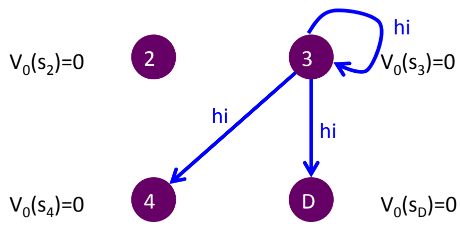
Theorem: optimal policy is
- Stationary (function only of last state, eg Markov)
- Time-independent
- Deterministic

for infinite / indefinite horizon problems

$$\pi^*(s) = \text{argmax}_{a \in Ap(s)} Q^*(s,a)$$
$$= \text{argmax}_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} Pr(s'|s,a) \left[\mathsf{R}(s,a,s') + \gamma V^*(s') \right]$$

Example: Value Iteration

$Q_1(s_{3,},hi)=.25*(4+0) + .25*(0+0) + .5*0 = 1$

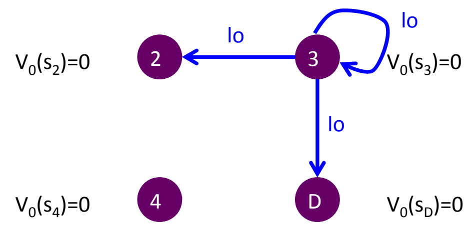


Example: Value Iteration

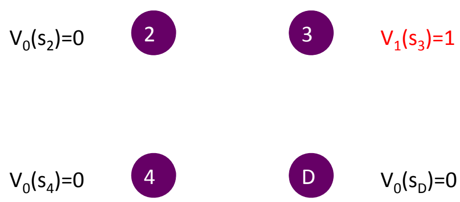
$Q_1(s_{3,},hi)=.25*(4+0) + .25*(0+0) + .5*0 = 1$

$Q_1(s_{3,},lo)=.5*(2+0) + .25*(0+0) + .25*0 = 1$

$V_1(s_3) = Max_a\, Q(s_3,a) = Max\ (1, 1) = 1$

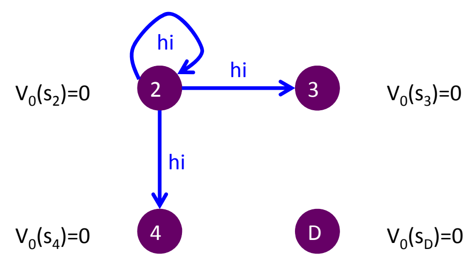


Example: Value Iteration



Example: Value Iteration

$Q_1(s_{2,},hi)=.5*(0+0) + .25*(3+0) + .25*(4+0) = 1$

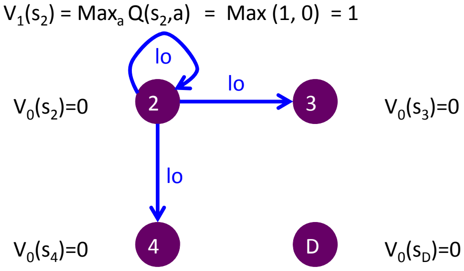


Example: Value Iteration

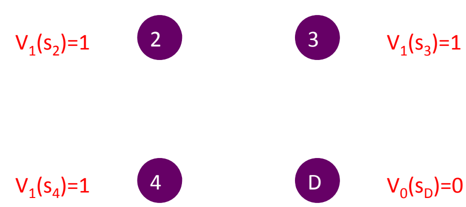
$Q_1(s_{2,},hi)=.5*(0+0) + .25*(3+0) + .25*(4+0) = 1$

$Q_1(s_{2,},lo)=.5*(0+0) + .25*(0) + .25*(0) = 0$

$V_1(s_2) = Max_a\, Q(s_2,a) = Max\ (1, 0) = 1$

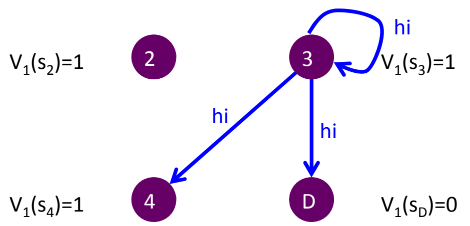


Example: Value Iteration



Round 2

$Q_2(s_3,hi)=.25*(4+1) + .25*(0+1) + .5*0 = 1.5$

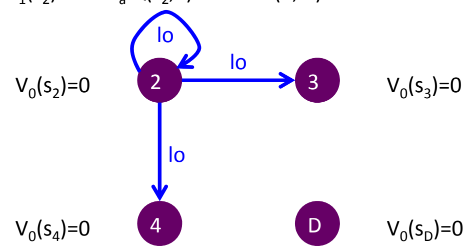


Round 2

$Q_1(s_2,hi)=.5*(0+0) + .25*(3+0) + .25*(4+0) = 1$

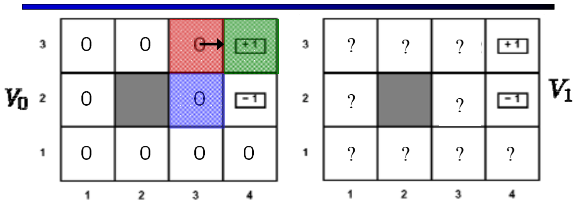
$Q_1(s_2,lo)=.5*(0+0) + .25*(0) + .25*(0) = 0$

$V_1(s_2) = \text{Max}_a\, Q(s_2,a) = \text{Max}(1, 0) = 1$



Example: Bellman Updates

Example: γ=0.9, living reward=0, noise=0.2

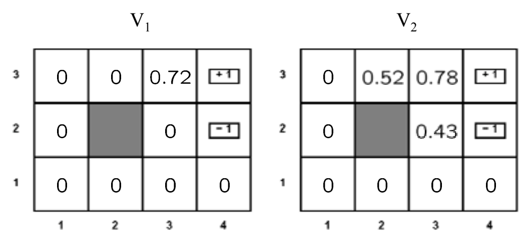


$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V_i(s')\right] = \max_a Q_{i+1}(s,a)$$

$$Q_1(\langle 3,3\rangle, \text{right}) = \sum_{s'} T(\langle 3,3\rangle, \text{right}, s')\left[R(\langle 3,3\rangle, \text{right}, s') + \gamma V_i(s')\right]$$

$$= 0.8*[0.0+0.9*1.0] + 0.1*[0.0+0.9*0.0] + 0.1*[0.0+0.9*0.0]$$

Example: Value Iteration



- Information propagates outward from terminal states and eventually all states have correct value estimates

Convergence

- Define the max-norm: $||U|| = \max_s |U(s)|$
- Theorem: For any two approximations U and V
  $$||U^{t+1} - V^{t+1}|| \le \gamma\, ||U^t - V^t||$$
  - I.e. any distinct approximations must get closer to each other, so, in particular, any approximation must get closer to the true U and value iteration converges to a unique, stable, optimal solution
- Theorem:
  $$||U^{t+1} - U^t|| < \epsilon, \Rightarrow ||U^{t+1} - U|| < 2\epsilon\gamma/(1-\gamma)$$
  - I.e. once the change in our approximation is small, it must also be close to correct

Value Iteration Complexity

- Problem size:
  - |A| actions and |S| states
- Each Iteration
  - Computation: $O(|A|\cdot|S|^2)$
  - Space: $O(|S|)$
- Num of iterations
  - Can be exponential in the discount factor γ

Summary: Value Iteration

- Idea:
  - Start with $V_0^*(s) = 0$, which we know is right (why?)
  - Given V_i^*, calculate the values for all states for depth i+1:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

  - This is called a value update or Bellman update
  - Repeat until convergence
- Theorem: will converge to unique optimal values
  - Basic idea: approximations get refined towards optimal values
  - ***Policy may converge*** long before values do

Asynchronous Value Iteration

- States may be backed up in any order
  - instead of an iteration by iteration
- As long as all states backed up infinitely often
  - Asynchronous Value Iteration converges to optimal

Asynch VI: Prioritized Sweeping

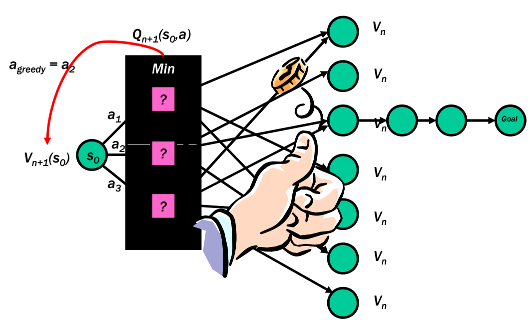
- Why backup a state if values of successors same?
- Prefer backing a state
  - whose successors had most change
- Priority Queue of (state, expected change in value)
- Backup in the order of priority
- After backing a state update priority queue
  - for all predecessors

Asynchonous Value Iteration Real Time Dynamic Programming

[Barto, Bradtke, Singh'95]

- Trial: simulate greedy policy starting from start state; perform Bellman backup on visited states
- RTDP: repeat Trials until value function converges

RTDP Trial



Comments

- Properties
  - if all states are visited infinitely often then $V_n \rightarrow V^*$
- Advantages
  - Anytime: more probable states explored quickly
- Disadvantages
  - complete convergence can be slow!