CSE 473

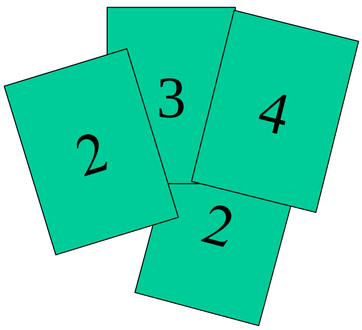
Lecture 16

Markov Decision Processes (MDPs) Part II

Last Time: High-Low as an MDP

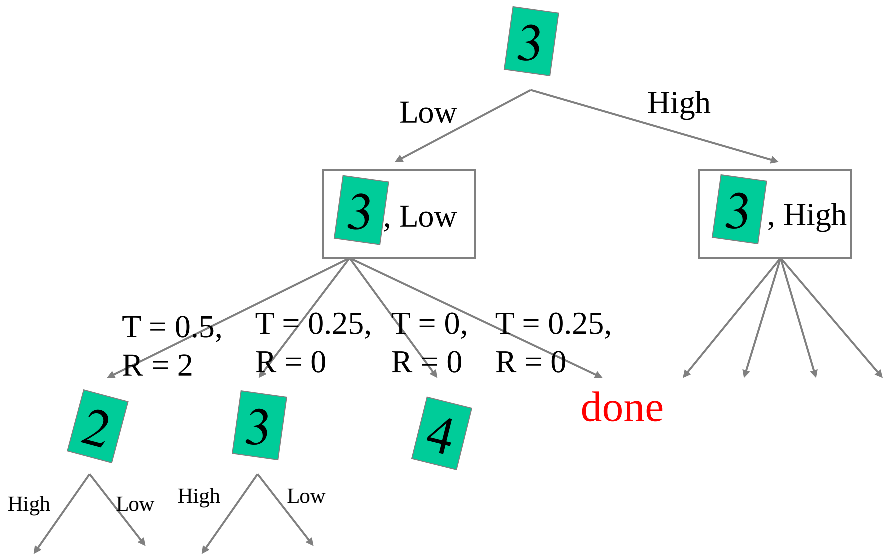
- States:
  - 2, 3, 4, done
- Actions:
  - High, Low
- Model: T(s, a, s'):
  - P(s'=4 | 4, Low) = 1/4
  - P(s'=3 | 4, Low) = 1/4
  - P(s'=2 | 4, Low) = 1/2
  - P(s'=done | 4, Low) = 0
  - P(s'=4 | 4, High) = 1/4
  - P(s'=3 | 4, High) = 0
  - P(s'=2 | 4, High) = 0
  - P(s'=done | 4, High) = 3/4
  - …

Twice as many 2's



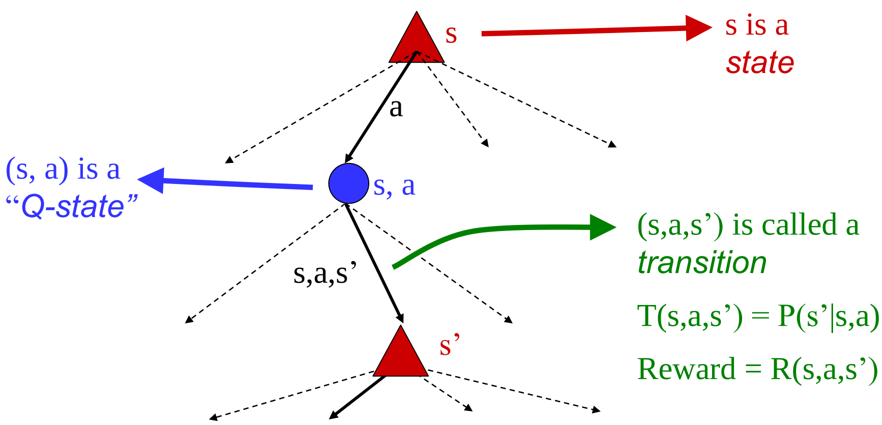
- Rewards: R(s, a, s'):
  - Number shown on s' if $s' < s \wedge a=$"Low" etc.
  - 0 otherwise
- Start: 3

Search Tree: High-Low



MDP Search Trees

- Each MDP state gives an expectimax-like search tree



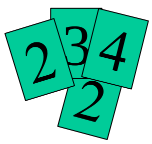
Utilities of Reward Sequences

- What is an "optimal" policy?
  - Each transition s,a,s' produces a reward (+ve, -ve, or 0)
  - Need to define utility of a *sequence* of rewards

- Idea 1:
  - Additive utility:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$$

Defining Utilities



- Problem: Infinite state sequences have infinite total reward
- Solutions:
  - Impose a *Finite Horizon* (deadline):
    - Terminate episodes after a fixed T steps (e.g. life)
    - Gives nonstationary policies (π depends on time left)
  - Absorbing state: guarantee that a terminal state will eventually be reached (like "done" for High-Low)
  - Discounting: Make infinite sum finite using γ ($0 < \gamma < 1$)

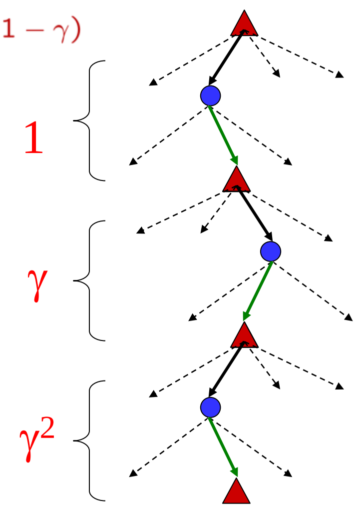
$$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

Discounting Rewards

$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

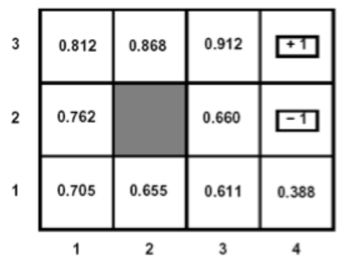
- Typically discount rewards by $\gamma < 1$ each time step
  - Sooner rewards have higher utility than later rewards
  - Also helps the algorithms converge



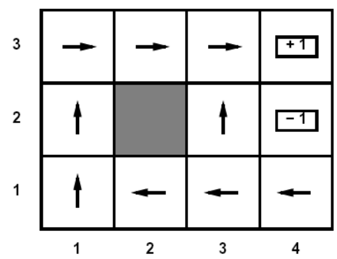
Optimal Utilities and Policy

- Define the value of a state s:
  $V^*(s)$ = expected utility starting in s and acting optimally
- Define the value of a Q-state (s,a):
  $Q^*(s,a)$ = expected utility starting in s, taking action a and thereafter acting optimally
- Define the optimal policy:
  $\pi^*(s)$ = optimal action from state s

Values



Optimal Policy



Bellman Equation

Richard Bellman
(1920-1984)

- Simple one-step look-ahead *recursive* relationship between optimal utility values
- Start with:

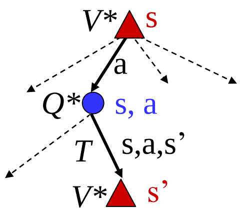
$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

- Combine to get Bellman Equation:

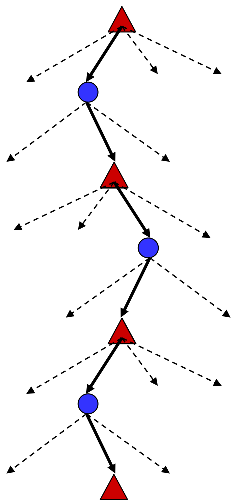
$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

recursive



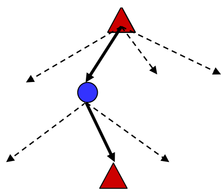
Why not use Expectimax?

- Problems:
  - The tree is usually infinite
  - Same states appear over and over
  - Need to search once for each state
- Idea: Value iteration
  - Compute optimal values for all states all at once iteratively (using successive approximations)
  - Bottom-up dynamic programming
  - Simple table look-up for any state



Value Iteration Idea

- Calculate estimates $V_k^*(s)$
  - The optimal value considering only *next k* time steps (next *k* rewards)
  - As $k \to \infty$, V_k approaches the optimal value
- Why should this work?
  - If discounting, distant rewards become negligible
  - If terminal states reachable from everywhere, fraction of episodes not ending becomes negligible
- Otherwise, can get infinite expected utility and this approach actually won't work



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

Example: Bellman Updates

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

V_0

3	0	0	0	+1
2	0		0	−1
1	0	0	0	0
	1	2	3	4

V_1

3	?	?	?	+1
2	?		?	−1
1	?	?	?	?
	1	2	3	4

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

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

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

= 0.72

Example: Value Iteration

V_1

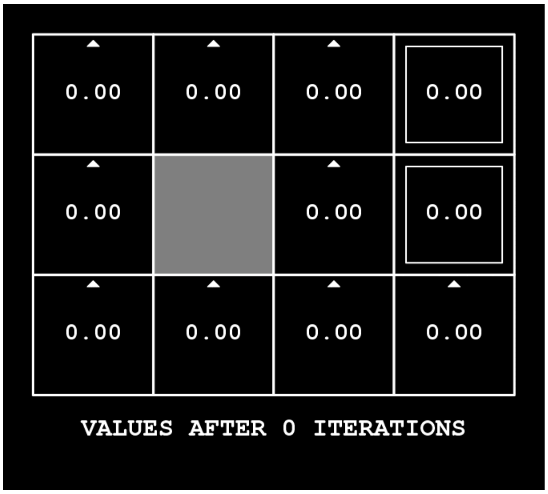
3	0	0	0.72	+1
2	0		0	−1
1	0	0	0	0
	1	2	3	4

V_2

3	0	0.52	0.78	+1
2	0		0.43	−1
1	0	0	0	0
	1	2	3	4

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

Example: Value Iteration (Movie)



Next Time

- Finding the optimal policy
- Reinforcement Learning
- To Do
  - Read chapters 17 and 21