Quiz 7: MDPs

- MDPs must have terminal states. **False**
- Every MDP is also a search problem. **False**
- Every search problem can be phrased as an MDP. **True**
- “Markov” means the future is independent of the past, given the present. **True**
- There are many (>17) ways to define stationary utilities over sequences of rewards. **False**
- Discounting with $0<\gamma<1$ can lead to infinite sequence rewards, even if every reward is bounded by some R. **False**

CS 511a: Artificial Intelligence
Fall 2013

Lecture 9: MEU / Markov Processes
10/10/2013

Kilian Weinberger

Many slides over the course adapted from either Dan Klein, Stuart Russell or Andrew Moore

Announcements

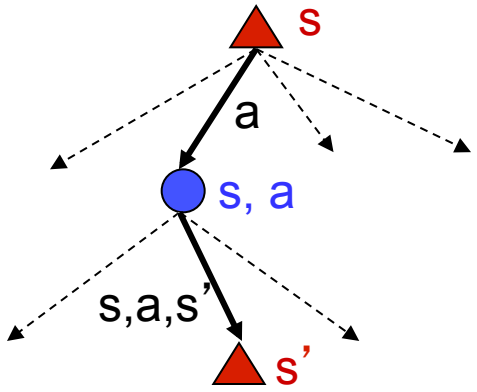
- HW 2 due Tuesday.
- Project 3 out today!

- Should we move Midterm to Oct 29th?

- Read: **Ch. 17.1-3, S&B Ch. 6.1,2,5**

Recap: Defining MDPs

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

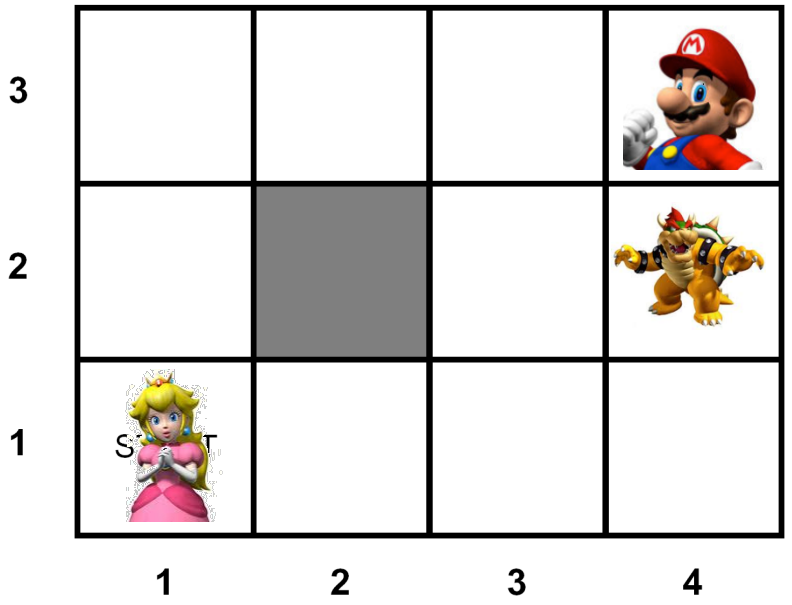


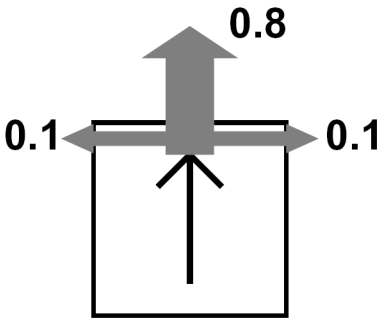
- MDP quantities so far:
  - Policy = Choice of action for each state
  - Utility (or return) = sum of discounted rewards

- **Quiz:** What is different from our previous definition of search problems?

Grid World

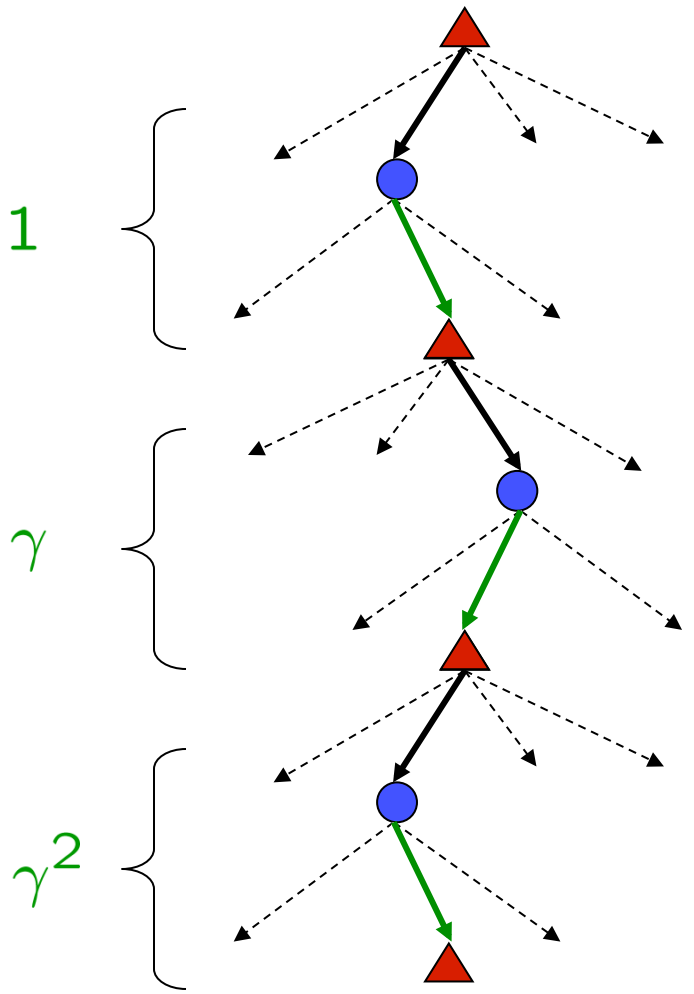
- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Small (negative) "reward" each step
- Big rewards come at the end
- Goal: maximize sum of rewards*





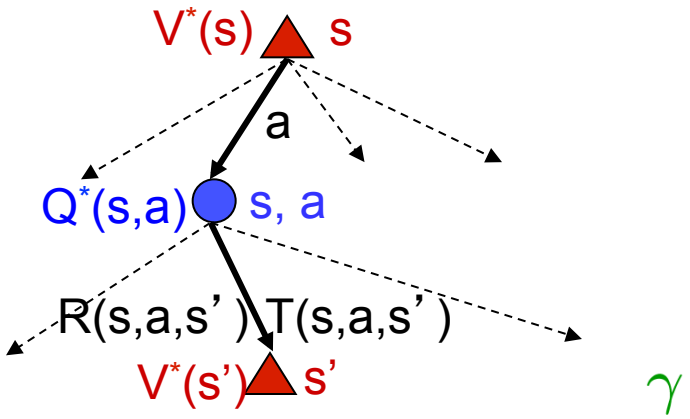
Discounting

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

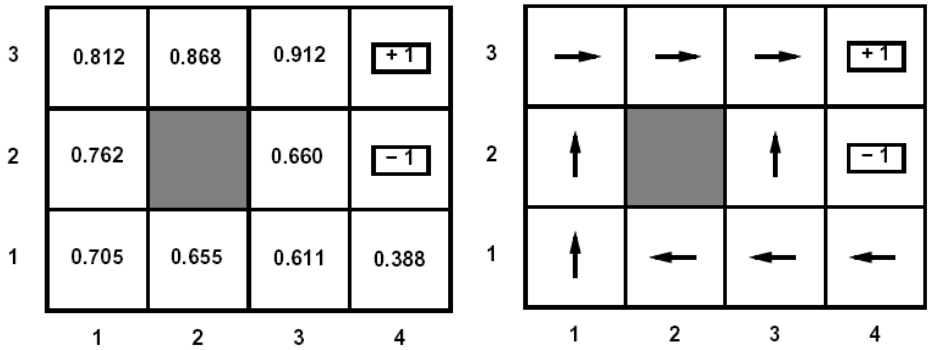


Optimal Utilities

- Fundamental operation: compute the values (optimal expectimax utilities) of states s
- Why? Optimal values define optimal policies!
- 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



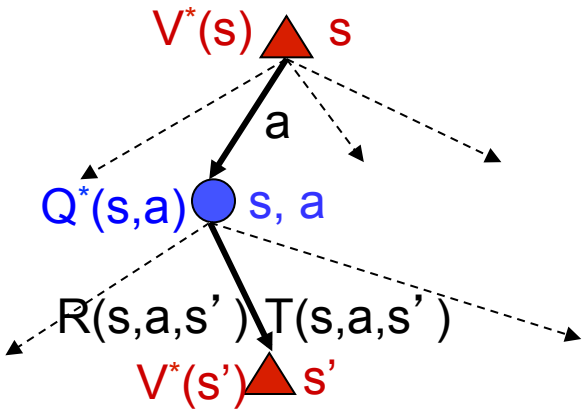
Quiz: Define V* in terms of Q* and vice versa!



The Bellman Equations

- Definition of "optimal utility" leads to a simple one-step lookahead relationship amongst optimal utility values:

  Optimal rewards = maximize over first action and then follow optimal policy



- Formally:

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

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

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

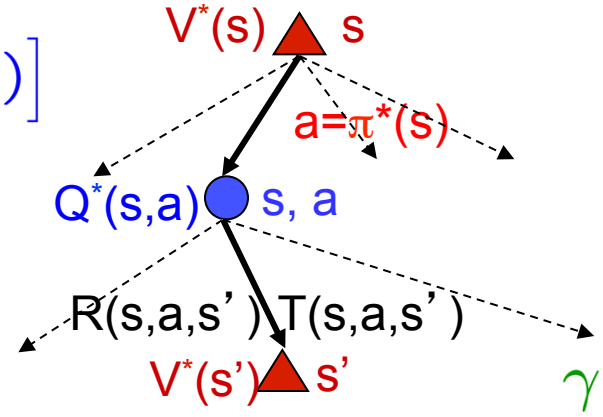
Solving MDPs

- We want to find the optimal policy π^*
- Proposal 1: modified expectimax search, starting from each state s:

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

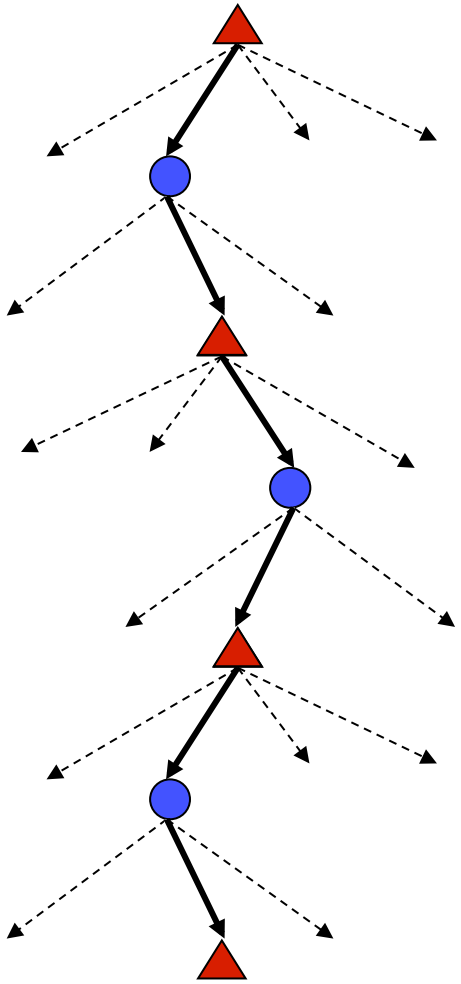
$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

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

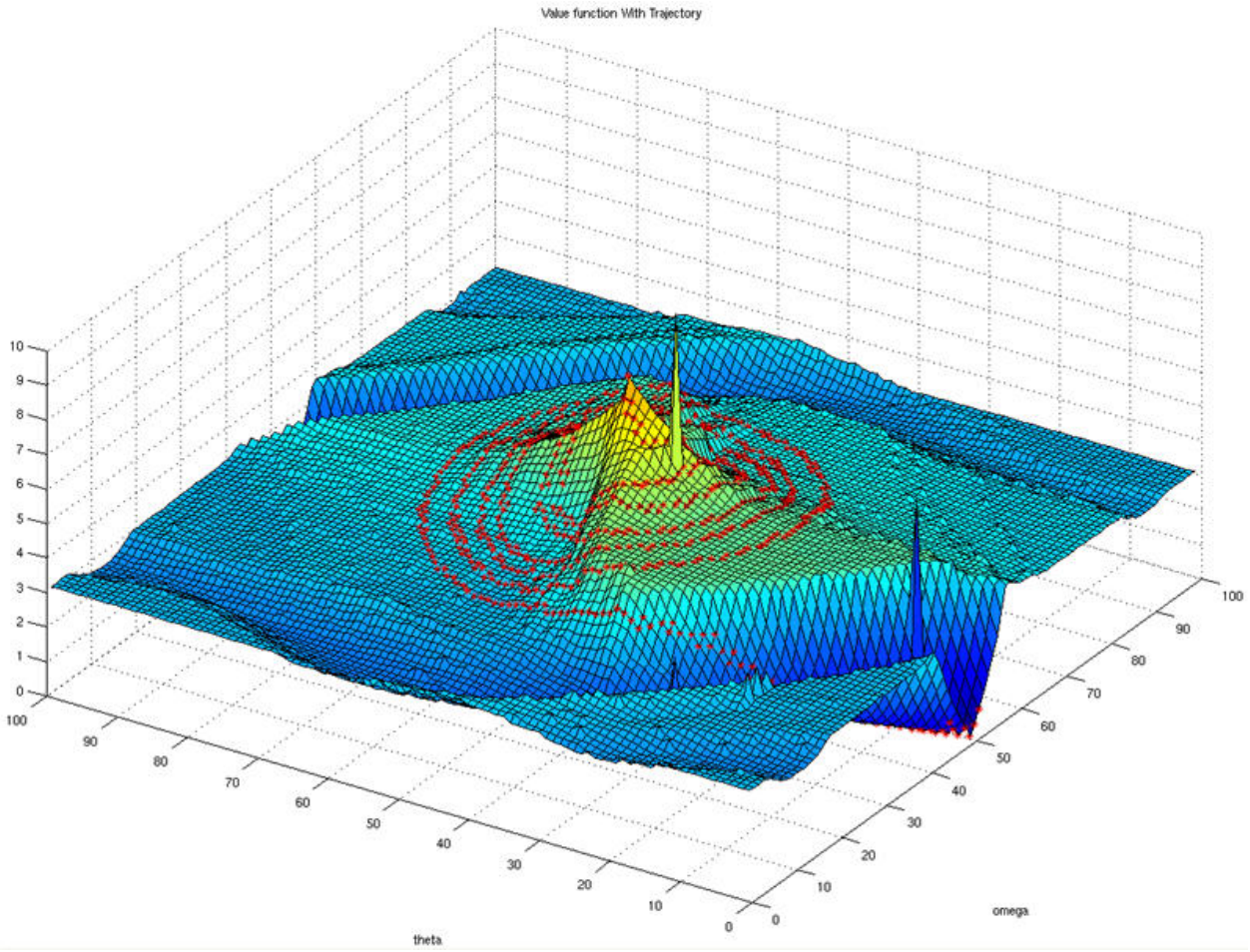


Why Not Search Trees?

- Why not solve with expectimax?
- Problems:
  - This tree is usually infinite (why?)
  - Same states appear over and over (why?)
  - We would search once per state (why?)
- Idea: Value iteration
  - Compute optimal values for all states all at once using successive approximations
  - Will be a bottom-up dynamic program similar in cost to memoization
  - Do all planning offline, no replanning needed!

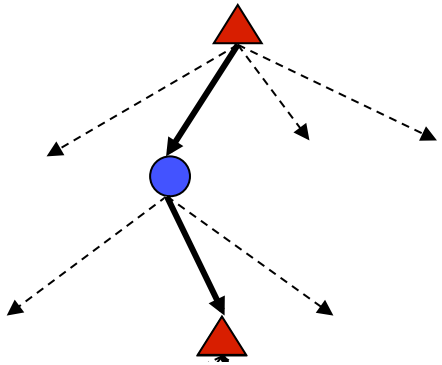


Value Iteration



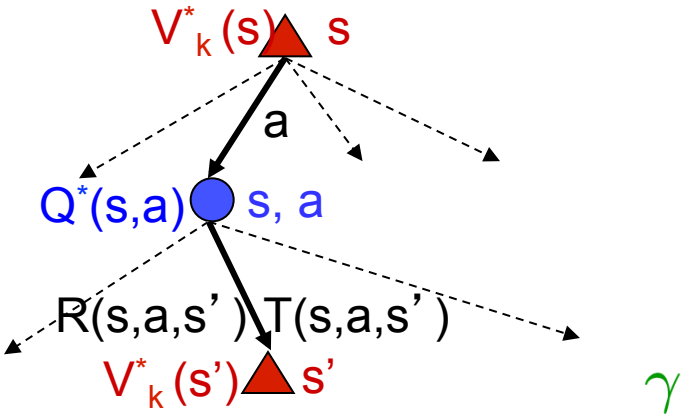
Value Estimates

- Calculate estimates $V_k^*(s)$
  - Not the optimal value of s!
  - The optimal value considering only next k time steps (k rewards)
  - As $k \rightarrow \infty$, it approaches the optimal value
  - Why:
    - 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 then this approach actually won't work



Quiz

1. What is $V^*_0(s)$?
2. Express $V^*_{k+1}(s)$ in terms of $V^*_k(s')$.



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:

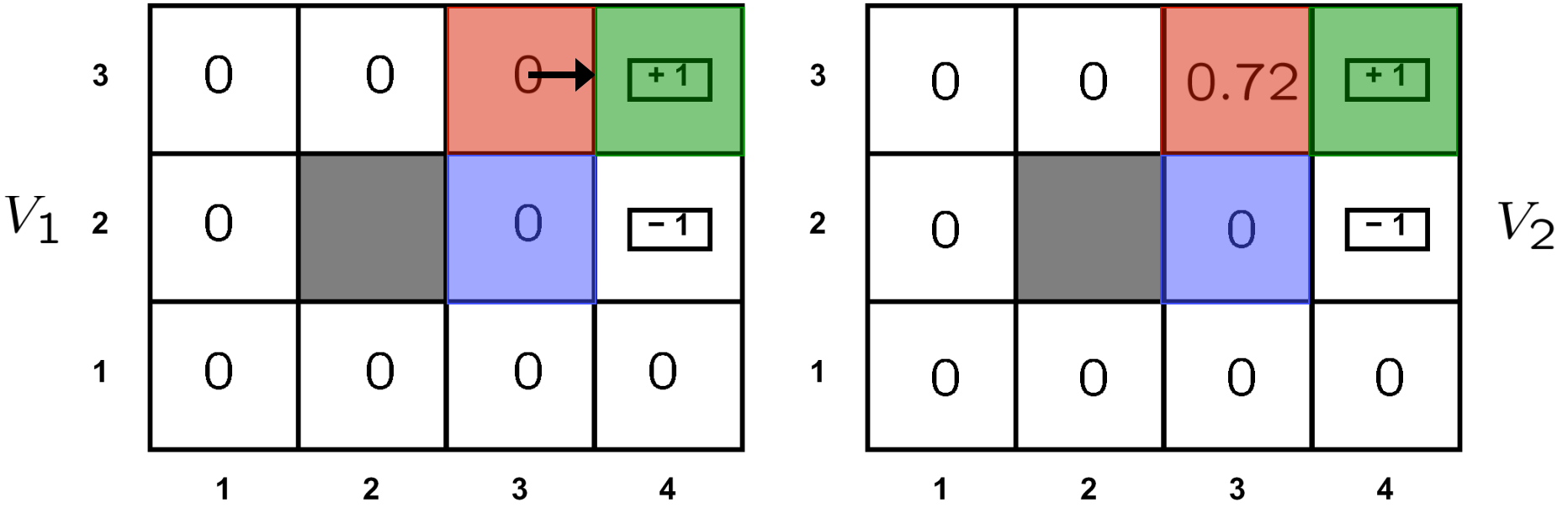
$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

  - 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

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

Example: Bellman Updates



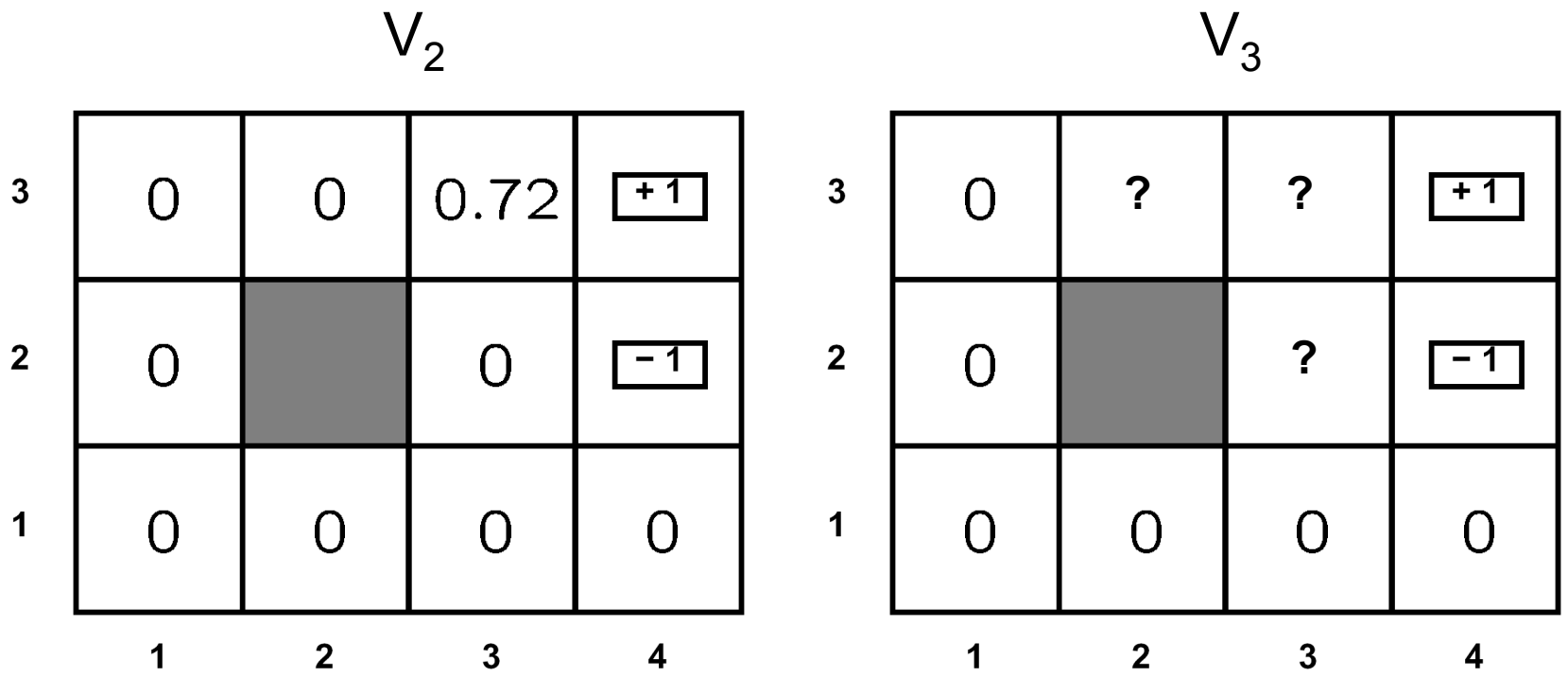
$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

$$V_2(\langle 3,3 \rangle) = \sum_{s'} T(\langle 3,3 \rangle, \text{right}, s') \left[R(\langle 3,3 \rangle) + 0.9 V_1(s') \right]$$

max happens for a=right, other actions not shown

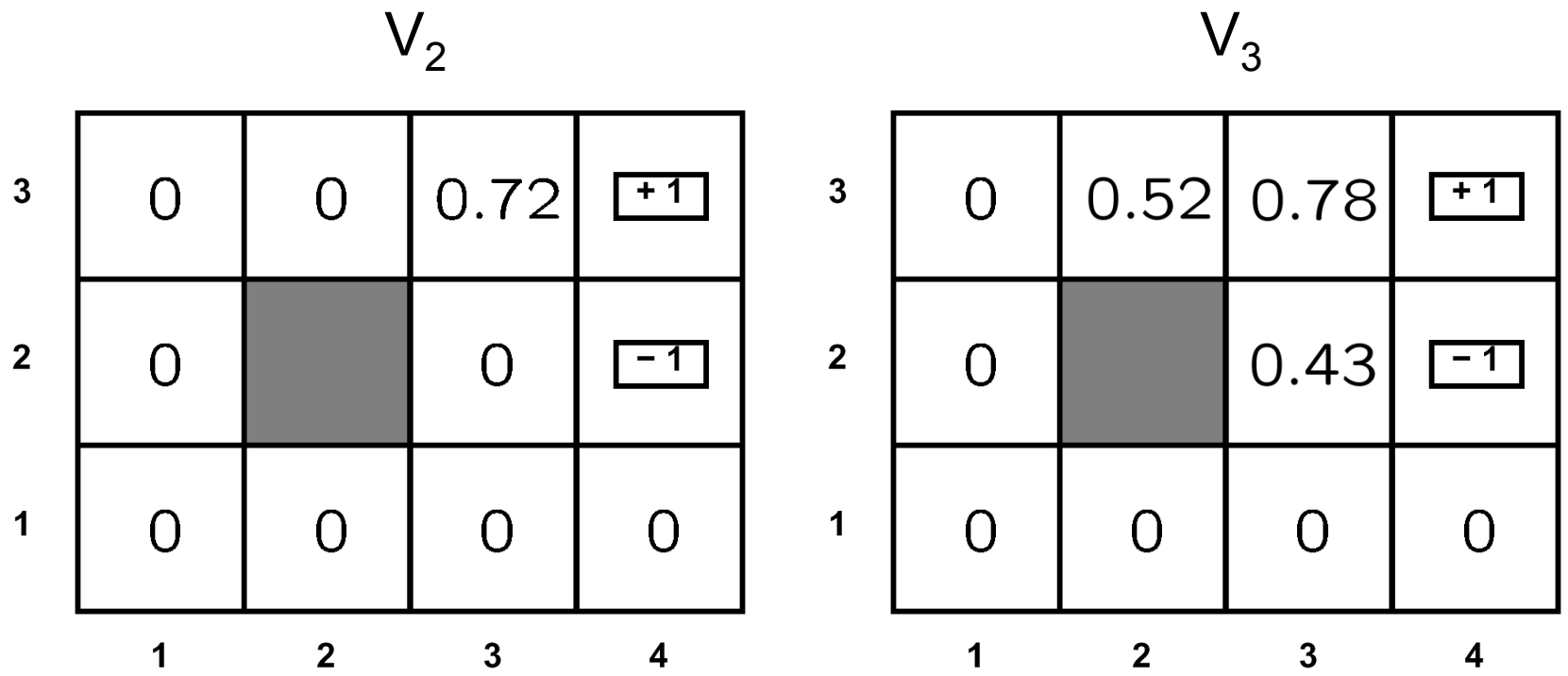
$$=0.8 \cdot [0+0.9 \cdot 1] + 0.1 \cdot [0+0.9 \cdot 0] + 0.1[0+0.9 \cdot 0]$$

Quiz: Value Iteration



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

Example: Value Iteration



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

Gridworld


Gridworld Display
0.00
0.00
0.72
1.00
0.00
0.00
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 2 ITERATIONS

Gridworld


Gridworld Display
0.00
0.52
0.78
1.00
0.00
0.43
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 3 ITERATIONS

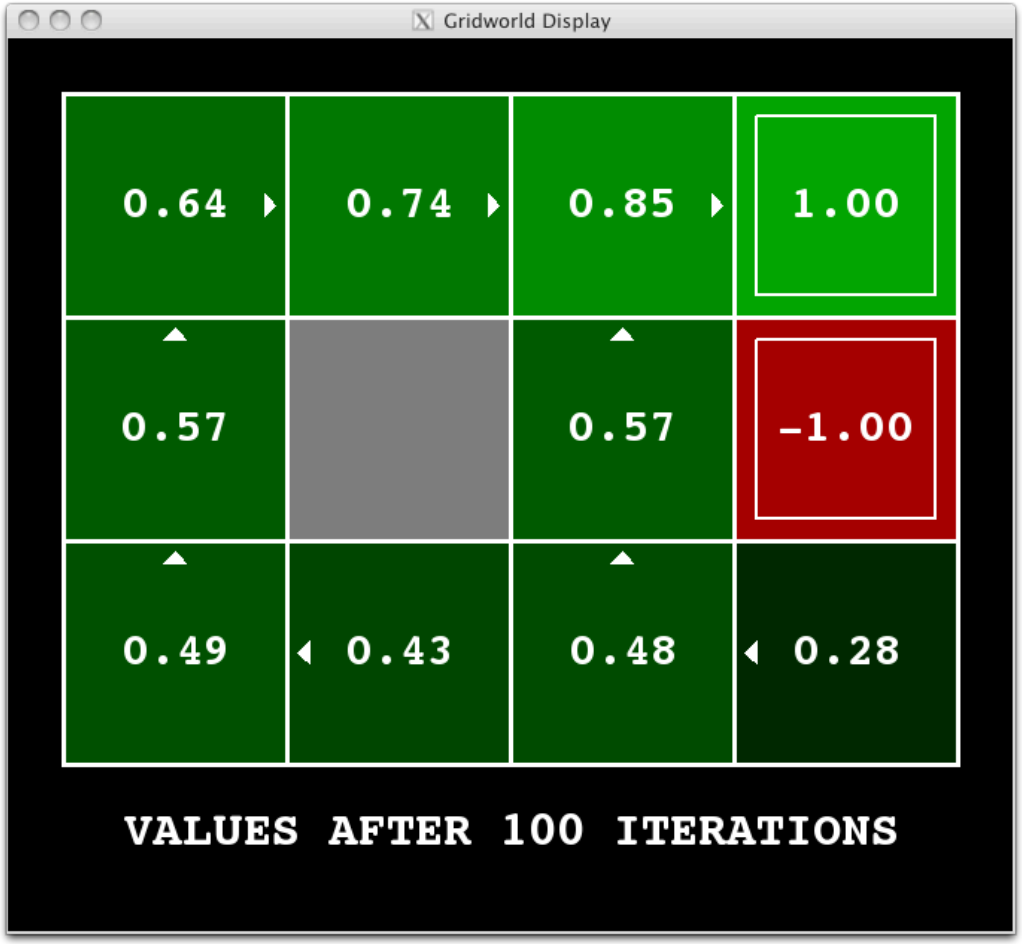
Gridworld


Gridworld Display
0.37
0.66
0.83
1.00
0.00
0.51
-1.00
0.00
0.00
0.31
0.00
VALUES AFTER 4 ITERATIONS

Gridworld


Gridworld Display
0.51
0.72
0.84
1.00
0.27
0.55
-1.00
0.00
0.22
0.37
0.13
VALUES AFTER 5 ITERATIONS

Gridworld


Gridworld Display
0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

Convergence*

- Define the max-norm: $||U|| = \max_s |U(s)|$
- Theorem: For any two approximations U and V

$$||U^{t+1} - V^{t+1}|| \leq \gamma\, ||U^t - V^t||$$

  - 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

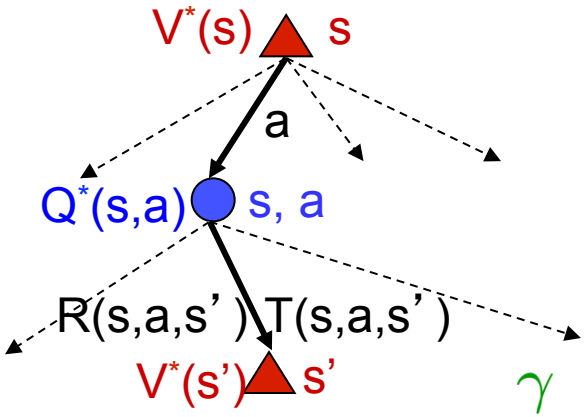
Practice: Computing Actions

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

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

  - Given optimal q-values Q*?

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



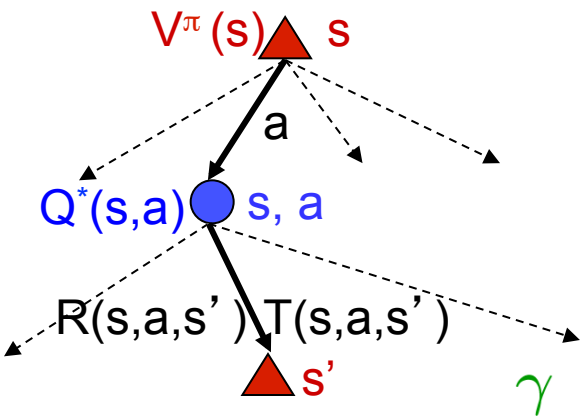
  - Lesson: actions are easier to select from Q's!

Policy Iteration

Utilities for Fixed Policies

- Another basic operation: compute the utility of a state s under a fix (general non-optimal) policy
- Define the utility of a state s, under a fixed policy π:

  $V^{\pi}(s)$ = expected total discounted rewards (return) starting in s and following π
- Recursive relation (one-step look-ahead / Bellman equation):



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

a=π(s)

$$V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^{\pi}(s')]$$

Policy Evaluation

- How do we calculate the V's for a fixed policy?
- Idea one: modify Bellman updates

$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_i^\pi(s')]$$

- Idea two: Optimal solution is stationary point (equality). Then it's just a linear system, solve with Matlab (or whatever)

Policy Iteration

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates:
  - Iterate until values converge

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

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s') \right]$$

Comparison

- In value iteration:
  - Every pass (or "backup") updates both utilities (explicitly, based on current utilities) and policy (possibly implicitly, based on current policy)
  - Policy might not change between updates (wastes computation) ☹
- In policy iteration:
  - Several passes to update utilities with frozen policy
  - Occasional passes to update policies
  - Value update can be solved as linear system
  - Can be faster, if policy changes infequently
- Hybrid approaches (asynchronous policy iteration):
  - Any sequences of partial updates to either policy entries or utilities will converge if every state is visited infinitely often

Asynchronous Value Iteration

- In value iteration, we update every state in each iteration
- Actually, *any* sequences of Bellman updates will converge if every state is visited infinitely often
- In fact, we can update the policy as seldom or often as we like, and we will still converge
- Idea: Update states whose value we expect to change:
  If $|V_{i+1}(s) - V_i(s)|$ is large then update predecessors of s