CPSC 4420/6420

Artificial Intelligence

06 – Markov Decision Processes

September 8, 2020

Announcements

- Project 1 is due tonight
- Quiz 2 is due on Thursday
- Sample exam questions will be posted later this week

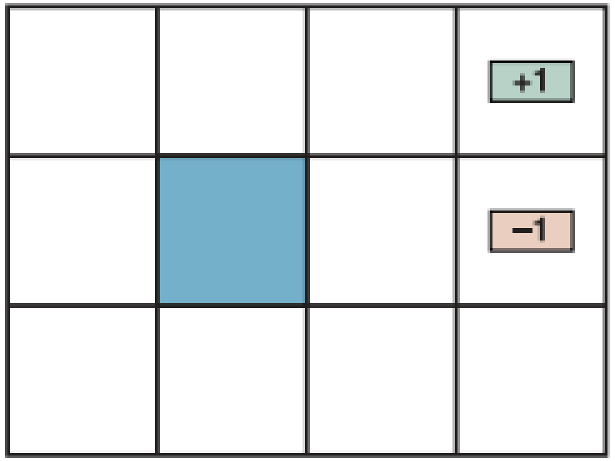
Lecture 6

Slide Credits:
Stuart Russell
Pieter Abbeel
Dan Klein
Ioannis Karamouzas

Markov Decision Process

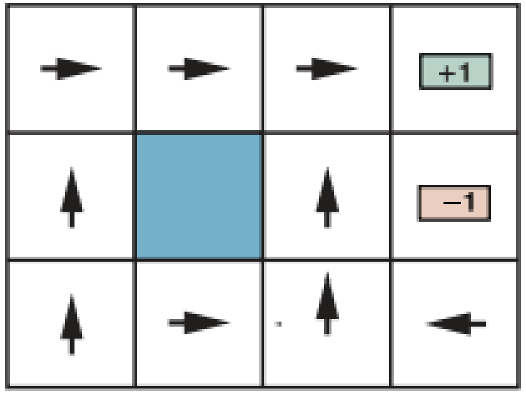
- An MDP is defined by:
  - A set of states S
  - A set of actions A
  - A transition function $T(s, a, s')$
    - Also called the model or the dynamics
    - Sometimes $P(s' | s, a)$
  - A reward function $R(s, a, s')$
    - Sometimes just $R(s)$ or $R(s')$
  - A start state s_0 (maybe a terminal one as well)

- MDPs are non-deterministic search problems



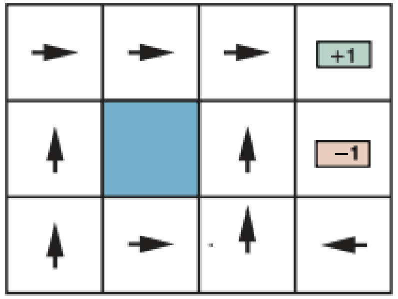
Policy

- In deterministic search problems, we seek an optimal sequence of actions (plan/path) from start to a goal
- For MDPs, we seek an optimal policy $\pi^*: S \rightarrow A$
  - A policy π gives an action for each state
  - An optimal policy, π^*, maximizes the *expected utility* if followed



Evaluating a policy

- Following a policy yields a random path
- The utility, U, of a policy is the (discounted) sum of the rewards along the path
  - Path 1, U = 1
  - Path 2, U = -1
  - Path 3, U = 1
  - ….



R(s) = -0.4

Discounting

- It's reasonable to maximize the sum of rewards
- It's also reasonable to prefer rewards now to rewards later
- Solution: values of rewards decay exponentially over time based on a *discount factor* $0 \leq \gamma \leq 1$

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

Discounting

- Example
  - U([1,2,3]) = 1*1 + γ*2 + γ^2*3 VS.
  - U([3,2,1]) = 1*3 + γ*2 + γ^2*1

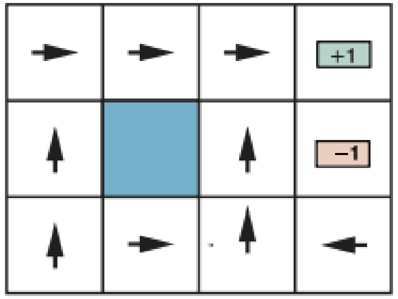
- Why discount?
  - Sooner rewards probably do have higher utility than later rewards
  - Also, helps algorithms converge

Avoiding infinite rewards

- Problem: If the game lasts forever, do we get infinite rewards?
- Solutions:
  - Finite horizon H
    - Terminate episodes after a fixed number of H steps
    - Gives nonstationary policies (π depends on time left)
  - Discounting: use $0 < \gamma < 1$

$$U([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = R_{\max}/(1-\gamma)$$

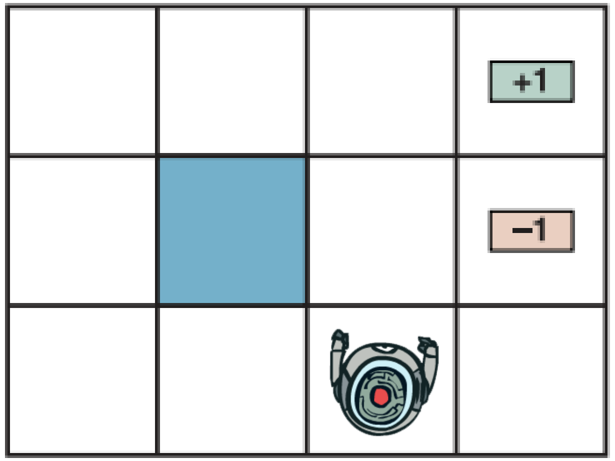
  - Absorbing state: guarantee that for every policy a terminal state will eventually be reached



R(s) = -0.4

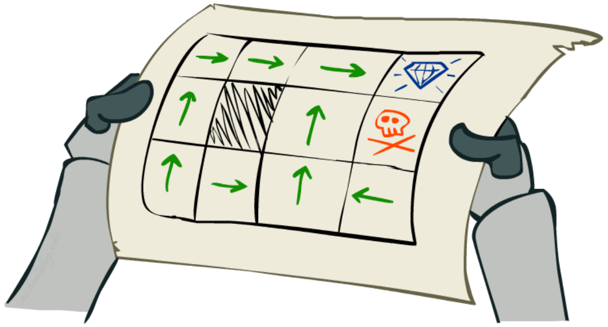
Revisiting MDPs

- An MDP is defined by:
  - A set of states S
  - A set of actions A
  - A transition function $P(s' \mid s, a)$ or $T(s, a, s')$
  - A reward function $R(s, a, s')$
  - A start state s_0 (maybe a terminal one as well)
  - Discount factor γ
  - Horizon H (can be infinite)
- MDP quantities so far
  - Policy π: Choice of action for each state
  - Utility $U_\pi = \sum_{t=0}^{H} \gamma^t R(\mathbf{s}_t)$: Sum of (discounted) rewards



Goal: $max_\pi \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(\mathbf{s}_t) \mid \pi\right]$

π*:

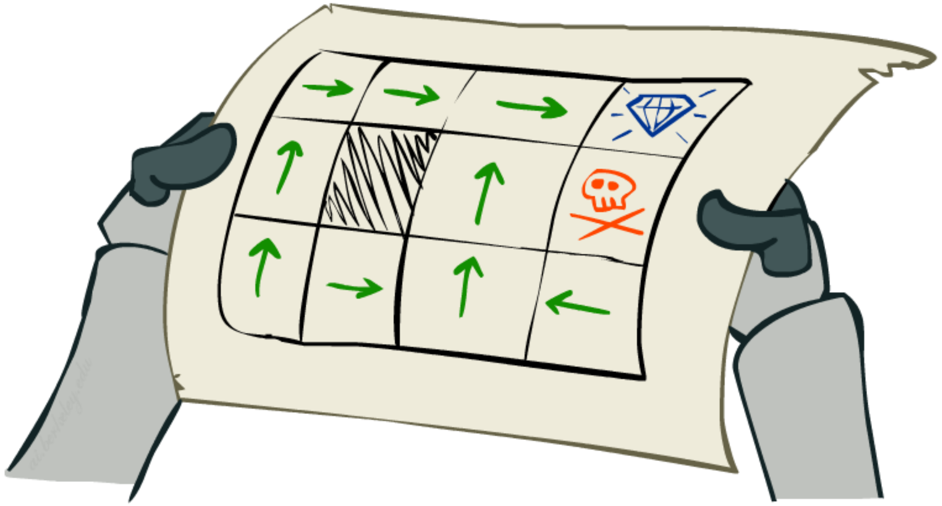


Solving MDPs

- Goal: find the optimal policy π*

$$\pi^* = \text{argmax}_\pi \, \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(\mathbf{s}_t) \mid \pi\right]$$

- Exact methods based on dynamic programming
  - Value Iteration
  - Policy Iteration

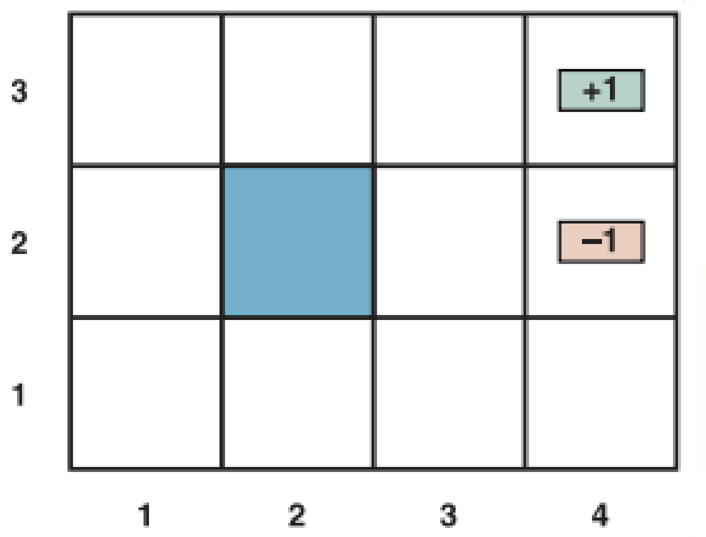


Solving MDPs: Value Iteration

*Optimal value function V**

$V^*(s)$ = sum of discounted rewards starting in s and acting optimally (for now)

<u>Let's assume:</u>
Noise = 0, γ = 1, living reward = 0



$V^*(4,3)$ =

$V^*(3,3)$ =

$V^*(2,3)$ =

$V^*(1,1)$ =

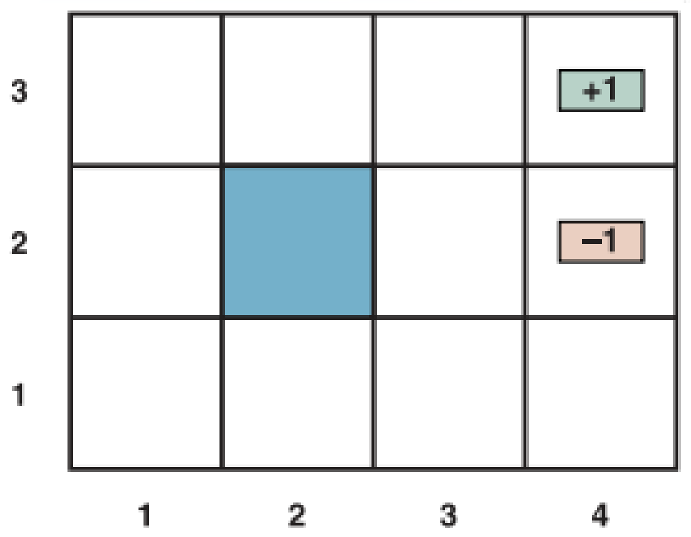
$V^*(4,2)$ =

*Optimal value function V**

$V^*(s)$ = sum of discounted rewards starting in s and acting optimally (for now)

<u>Let's assume:</u>

Noise = 0, γ = 1, living reward = 0



$V^*(4,3)$ = 1

$V^*(3,3)$ = 1

$V^*(2,3)$ = 1

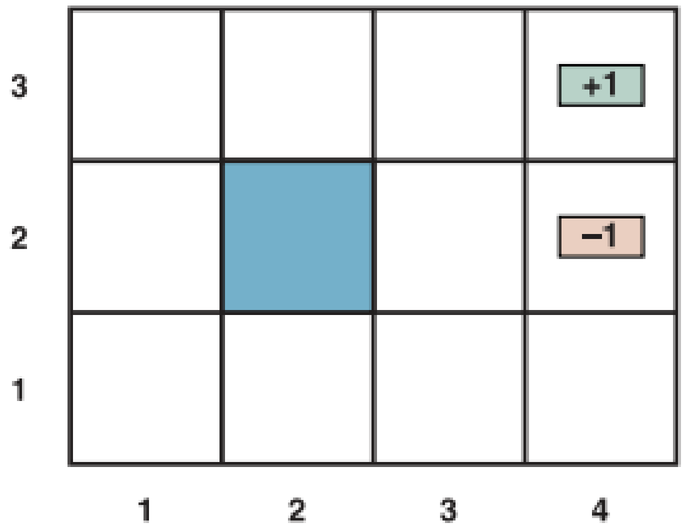
$V^*(1,1)$ = 1

$V^*(4,2)$ = -1

*Optimal value function V**

$V*(s)$ = sum of discounted rewards starting in s and acting optimally (for now)

<u>Let's assume:</u>
Noise = 0, γ = 0.9, living reward = 0



$V*(4{,}3)$ =

$V*(3{,}3)$ =

$V*(2{,}3)$ =

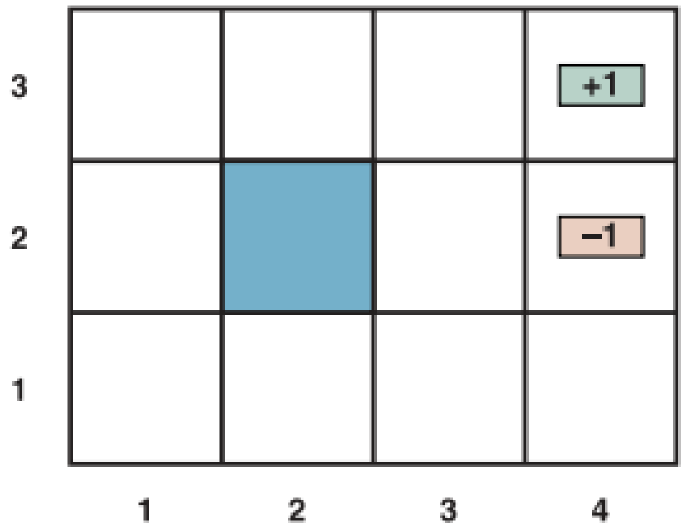
$V*(1{,}1)$ =

$V*(4{,}2)$ =

Optimal value function V^*

$V^*(s)$ = sum of discounted rewards starting in s and acting optimally (for now)

<u>Let's assume:</u>
Noise = 0, γ = 0.9, living reward = 0



$V^*(4,3)$ = 1

$V^*(3,3)$ = 0.9

$V^*(2,3)$ = 0.9×0.9 = 0.81

$V^*(1,1)$ = 0.9×0.9×0.9×0.9×0.9 = 0.59

$V^*(4,2)$ = -1

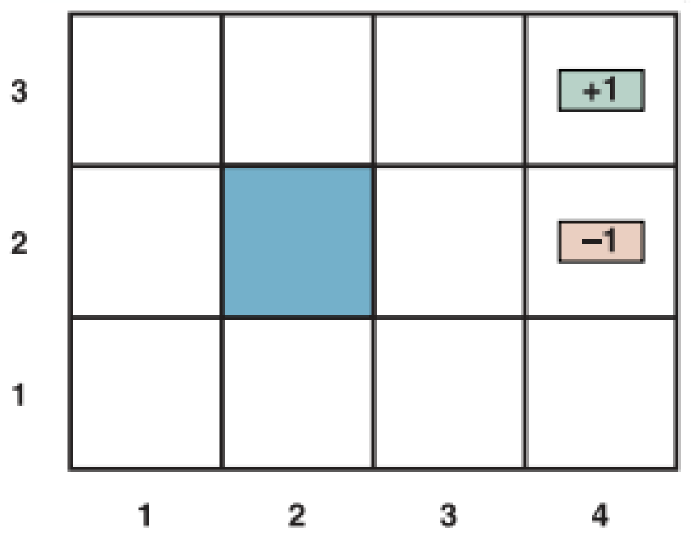
Optimal value function V^*

$V^*(s)$ = sum of discounted rewards starting in s and acting optimally (for now)

$$= \max_a \left(R(s,a,s') + \gamma V^*(s')\right)$$

<u>Let's assume:</u>

Noise = 0, γ = 0.9, living reward = 0



$V^*(4{,}3)$ = 1

$V^*(3{,}3)$ = 0.9 = 0 + $\gamma V^*(4{,}3)$ [action East]

$V^*(2{,}3)$ = 0.9×0.9 = 0.81 = 0 + $\gamma V^*(3{,}3)$ [action East]

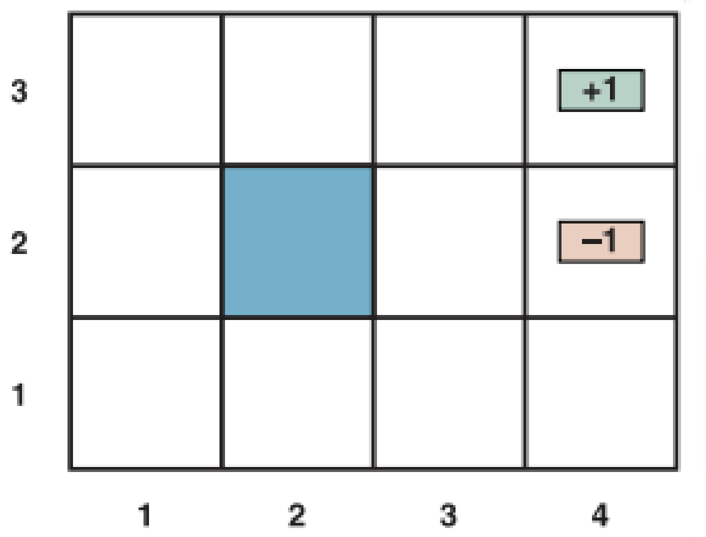
$V^*(1{,}1)$ = 0.9×0.9×0.9×0.9×0.9 = 0.59 = 0 + $\gamma V^*(1{,}2)$ [action North]

$V^*(4{,}2)$ = -1

*Optimal value function V**

<u>Let's assume:</u>

Noise = 0.2, γ = 0.9, living reward = 0



$V*(4,3)$ =

$V*(3,3)$ =

$V*(2,3)$ =

$V*(1,1)$ =

$V*(4,2)$ =

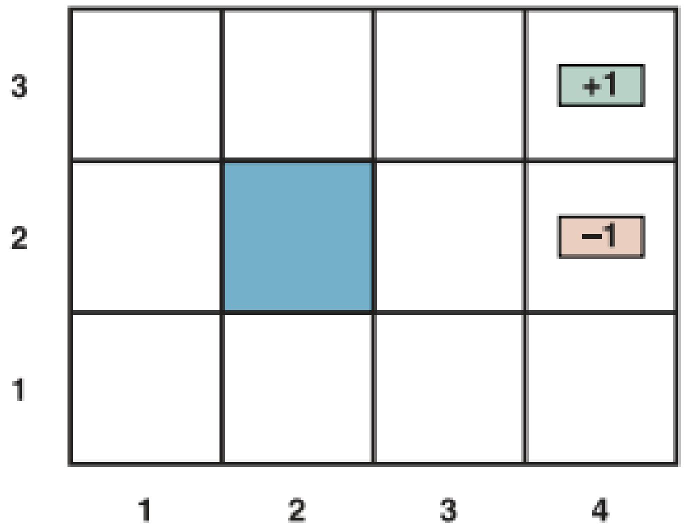
*Optimal value function V**

$V^*(s)$ = expected utility starting in s and acting optimally

$$= \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

<u>Let's assume:</u>

Noise = 0.2, γ = 0.9, living reward = 0



$V^*(4,3) = 1$

$V^*(3,3) = 0.8 \times 0.9 + 0.1 \times 0.9 \times V^*(3,3) + 0.1 \times 0.9 \times V^*(3,2)$ [action East]

$V^*(2,3) =$

$V^*(1,1) =$

$V^*(4,2) =$

Bellman equation

$$
\begin{aligned}
V^*(s) &= \max_a \mathbb{E}\Big[R(s,a,s') + \gamma V^*(s')\Big] \\
&= \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V^*(s'))
\end{aligned}
$$

Value iteration

- $V_0^*(s)$ = optimal value for state s when H=0
  - $V_0^*(s)$ = 0, $\forall$ s
- $V_1^*(s)$ = optimal value for state s when H=1
  - $V_1^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0^*(s'))$
- $V_2^*(s)$ = optimal value for state s when H=2
  - $V_2^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when H=k
  - $V_k^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$

Value iteration algorithm

function Value-Iteration (MDP)

Initialize $V_0^*(s) = 0, \forall s$

for k=1…H do

for each state s in S do

$$V_k^*(s) \leftarrow \max_a \sum P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

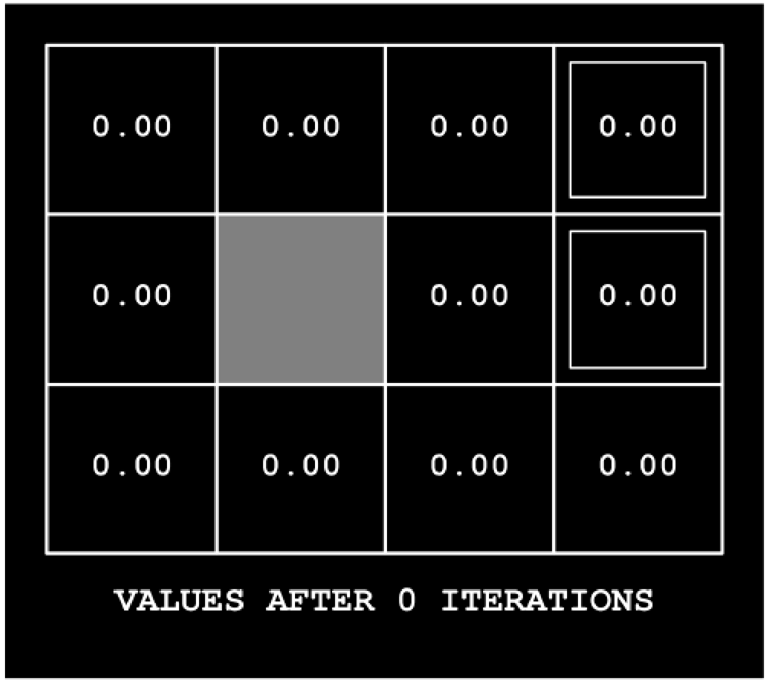
This is called a value update or Bellman update/back-up

Implementation: Use two vectors of size |S| storing value functions, one at time k and one at k-1

Value iteration example

$$V_0(s) \leftarrow 0$$

k = 0



Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_1(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 1

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 2

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 3

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 4

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 5

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 6

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 7

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 8

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 9

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 10

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 11

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 12

Noise = 0.2
Discount = 0.9
Living reward = 0

Value iteration example

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 100

Noise = 0.2
Discount = 0.9
Living reward = 0

Convergence of value iteration

Theorem. *Value iteration converges. At convergence, we have found the optimal value function V* for the discounted infinite horizon problem, which satisfies the Bellman equations*

$$\forall S \in S: \quad V^*(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

- Now we know how to act for infinite horizon with discounted rewards!
  - Run value iteration until convergence
  - This produces V*, which in turn tells you how to act:

    $$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

  - In fact policy may converge long before values do
- Complexity of each value iteration: $O(|S|^2|A|)$

Q-Values

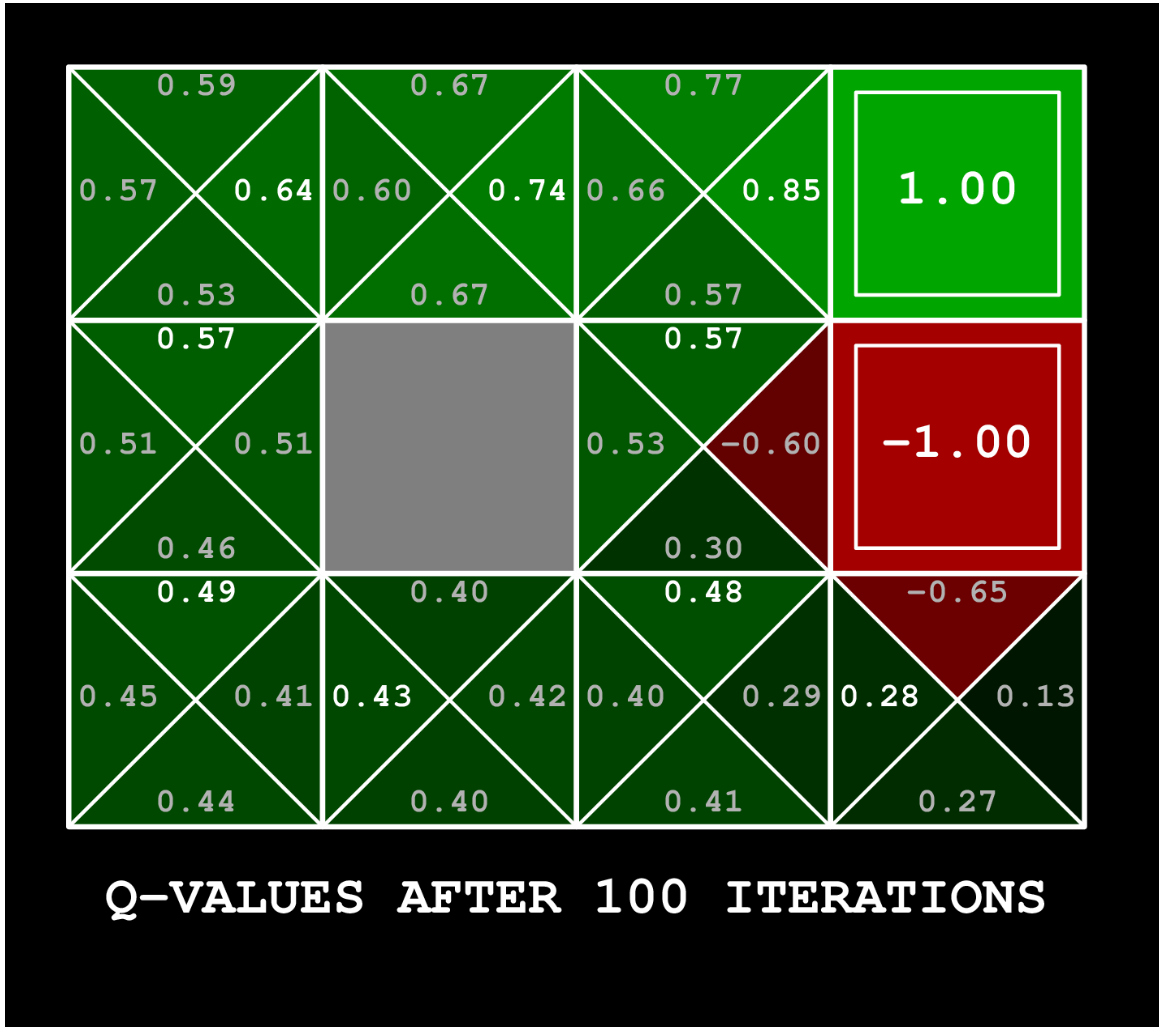
- Q*(s, a) = expected utility starting in s, taking action a and (thereafter) acting optimally
- Bellman equation

$$Q^*(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

- Q-Value Iteration

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

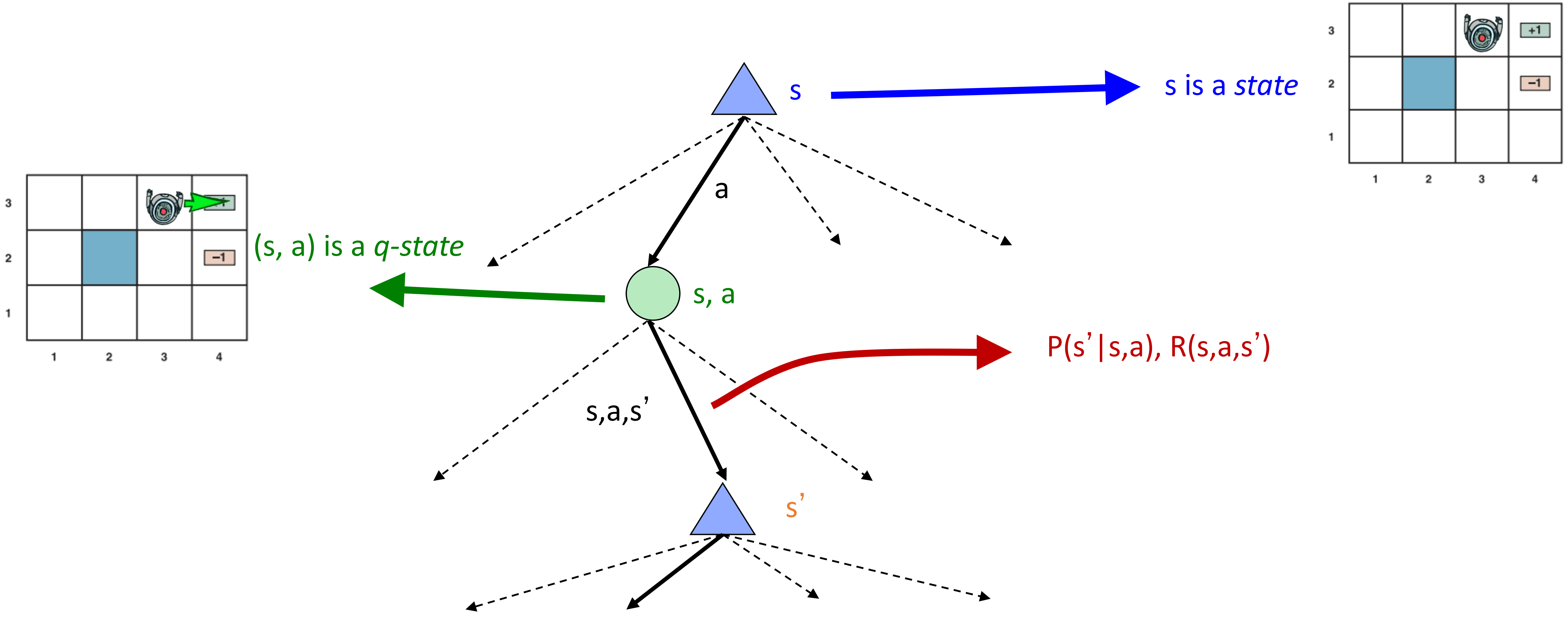
Gridworld: V* and Q*



Optimal utilities

- V*(s) = expected utility starting in s and acting optimally
- Q*(s,a) = expected utility starting out having taken action a from state s and (thereafter) acting optimally

MDP search tree



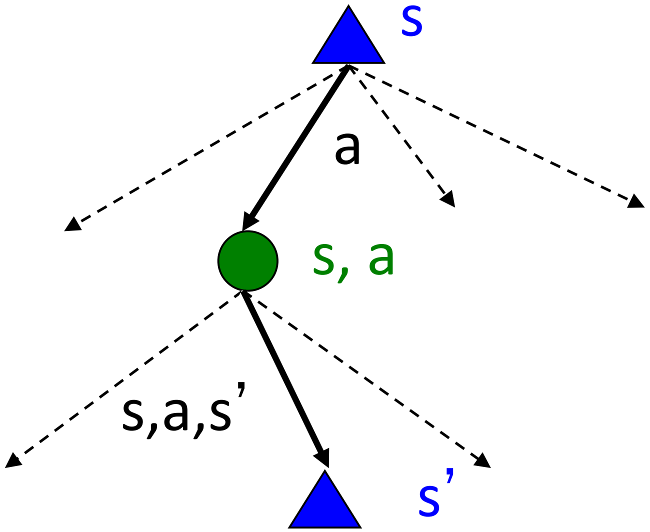
Bellman equations

- Recursive definition of values

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

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

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



- These are called the Bellman equations
  - Can be solved using one-step lookahead relationship amongst optimal values

Principle of maximum expected utility (MEU)

- A rational agent should chose the action that maximizes its expected utility
- The expected utility of an action is $EU(a|s) = \sum_{s'} P(s'|a,s)U(s')$