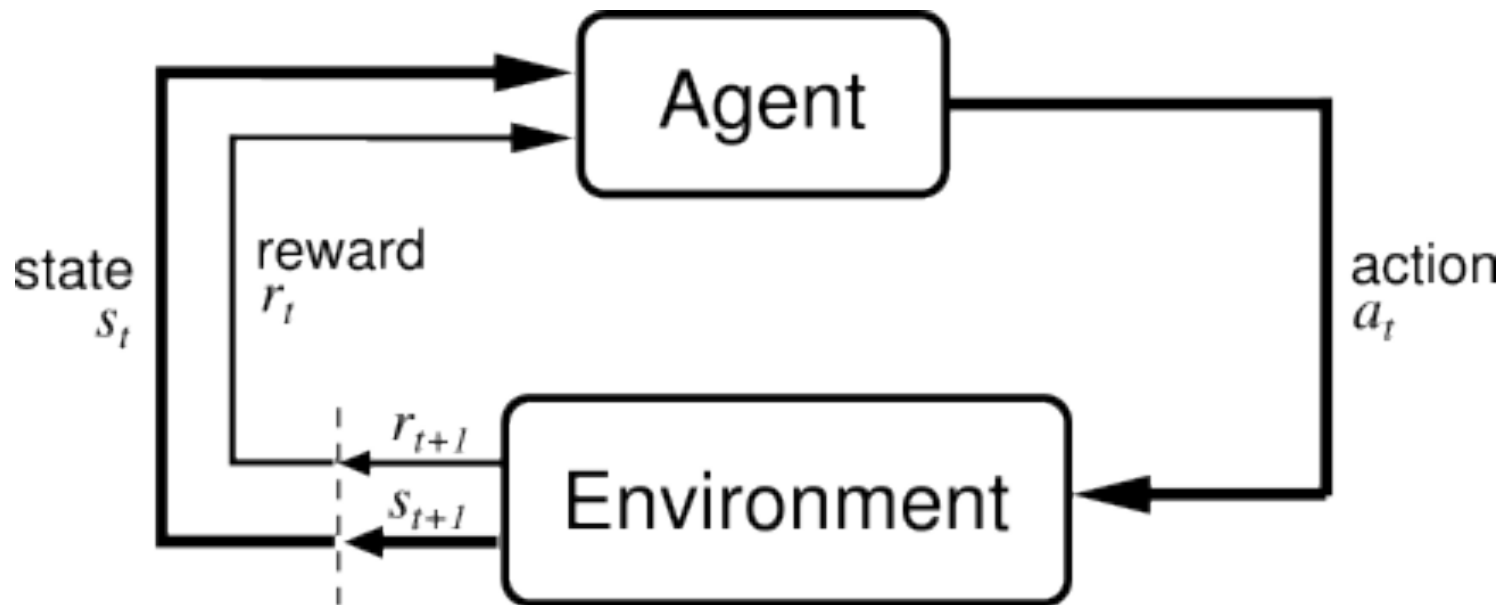


**Markov Decision Processes**  
**and**  
**Exact Solution Methods:**  
Value Iteration  
Policy Iteration  
Linear Programming

Pieter Abbeel  
UC Berkeley EECS

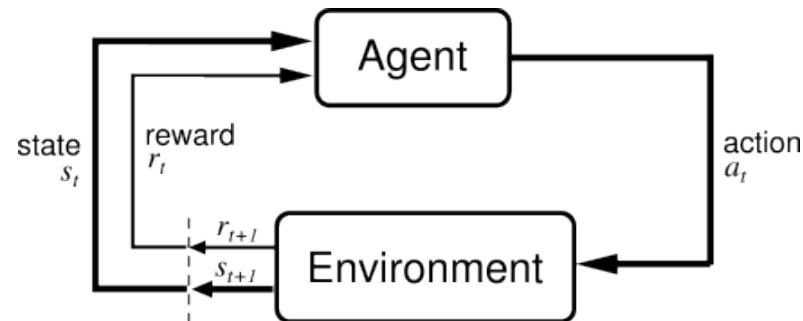
# Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

# Markov Decision Process (S, A, T, R, H)



Given

- S: set of states
- A: set of actions
- $T: S \times A \times S \times \{0, 1, \dots, H\} \rightarrow [0, 1]$ ,  $T_t(s, a, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- $R: S \times A \times S \times \{0, 1, \dots, H\} \rightarrow \mathbb{R}$ ,  $R_t(s, a, s') = \text{reward for } (s_{t+1} = s', s_t = s, a_t = a)$
- H: horizon over which the agent will act

Goal:

- Find  $\pi : S \times \{0, 1, \dots, H\} \rightarrow A$  that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg \max_{\pi} E \left[ \sum_{t=0}^H R_t(S_t, A_t, S_{t+1}) \mid \pi \right]$$

# Examples

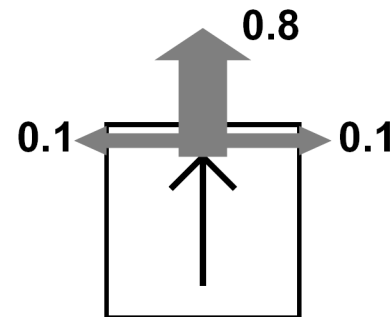
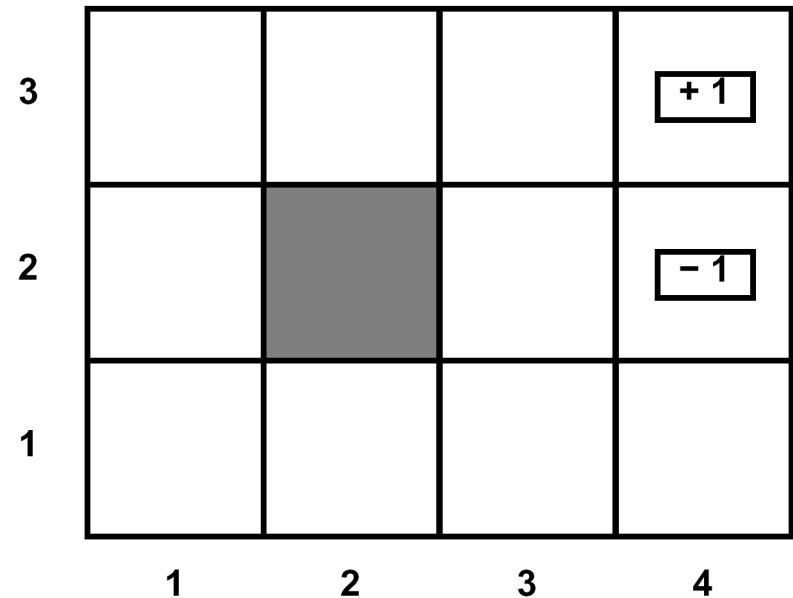
MDP (S, A, T, R, H),

goal:  $\max_{\pi} \mathbb{E} \left[ \sum_{t=0}^H R(S_t, A_t, S_{t+1}) | \pi \right]$

- ❑ Cleaning robot
- ❑ Walking robot
- ❑ Pole balancing
- ❑ Games: tetris, backgammon
- ❑ Server management
- ❑ Shortest path problems
- ❑ Model for animals, people

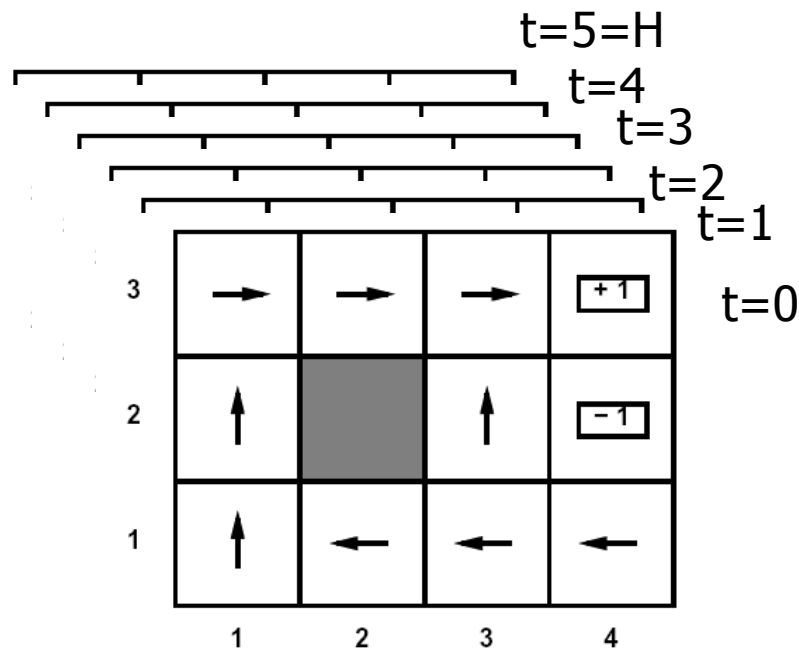
# Canonical Example: Grid World

- 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
- Big rewards come at the end



# Solving MDPs

- In an MDP, we want an optimal **policy**  $\pi^*: S \times 0:H \rightarrow A$ 
  - A policy  $\pi$  gives an action for each state for each time



- An optimal policy maximizes expected sum of rewards
- Contrast: In deterministic, want an optimal **plan**, or sequence of actions, from start to a goal

# Outline

- Optimal Control

=

given an MDP ( $S, A, T, R, \gamma, H$ )

find the optimal policy  $\pi^*$

- Exact Methods:

- ***Value Iteration***

- Policy Iteration

- Linear Programming

For now: discrete state-action spaces as they are simpler to get the main concepts across. Will consider continuous spaces later!

# Value Iteration

- Algorithm:

- Start with  $V_0^*(s) = 0$  for all  $s$ .

- For  $i=1, \dots, H$

- Given  $V_i^*$ , calculate for all states  $s \in S$ :

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

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

- $V_i^*(s)$  = the expected sum of rewards accumulated when starting from state  $s$  and acting optimally for a horizon of  $i$  steps



# Value Iteration in Gridworld

noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$

|        |        |        |           |
|--------|--------|--------|-----------|
| 0.00 ▶ | 0.00 ▶ | 0.00 ▶ | 1.00      |
| 0.00 ▶ |        | ◀ 0.00 | -1.00     |
| 0.00 ▶ | 0.00 ▶ | 0.00 ▶ | 0.00<br>▼ |

VALUES AFTER 1 ITERATIONS

# Value Iteration in Gridworld

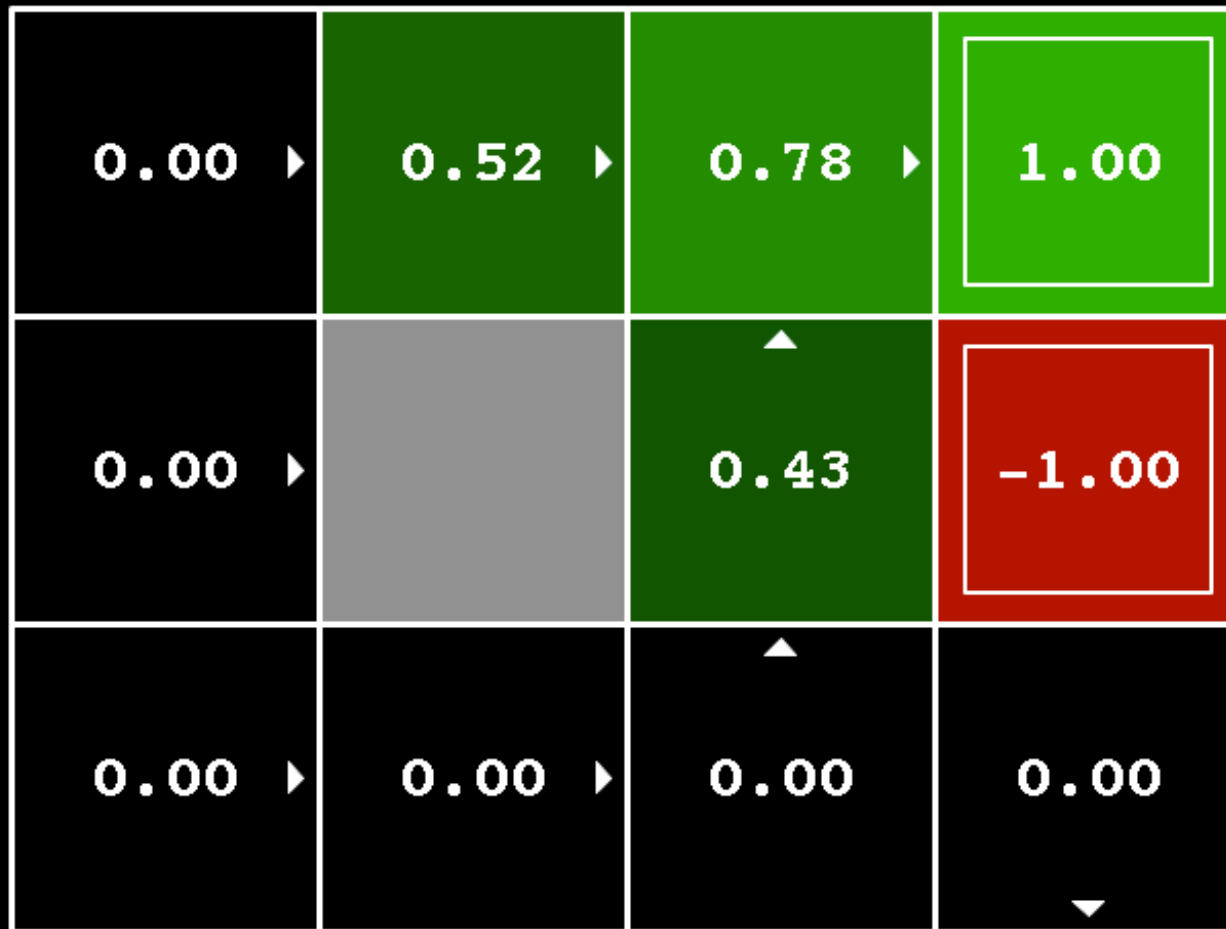
noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$

|        |        |           |           |
|--------|--------|-----------|-----------|
| 0.00 ▶ | 0.00 ▶ | 0.72 ▶    | 1.00      |
| 0.00 ▶ |        | ▲<br>0.00 | -1.00     |
| 0.00 ▶ | 0.00 ▶ | 0.00 ▶    | 0.00<br>▼ |

VALUES AFTER 2 ITERATIONS

# Value Iteration in Gridworld

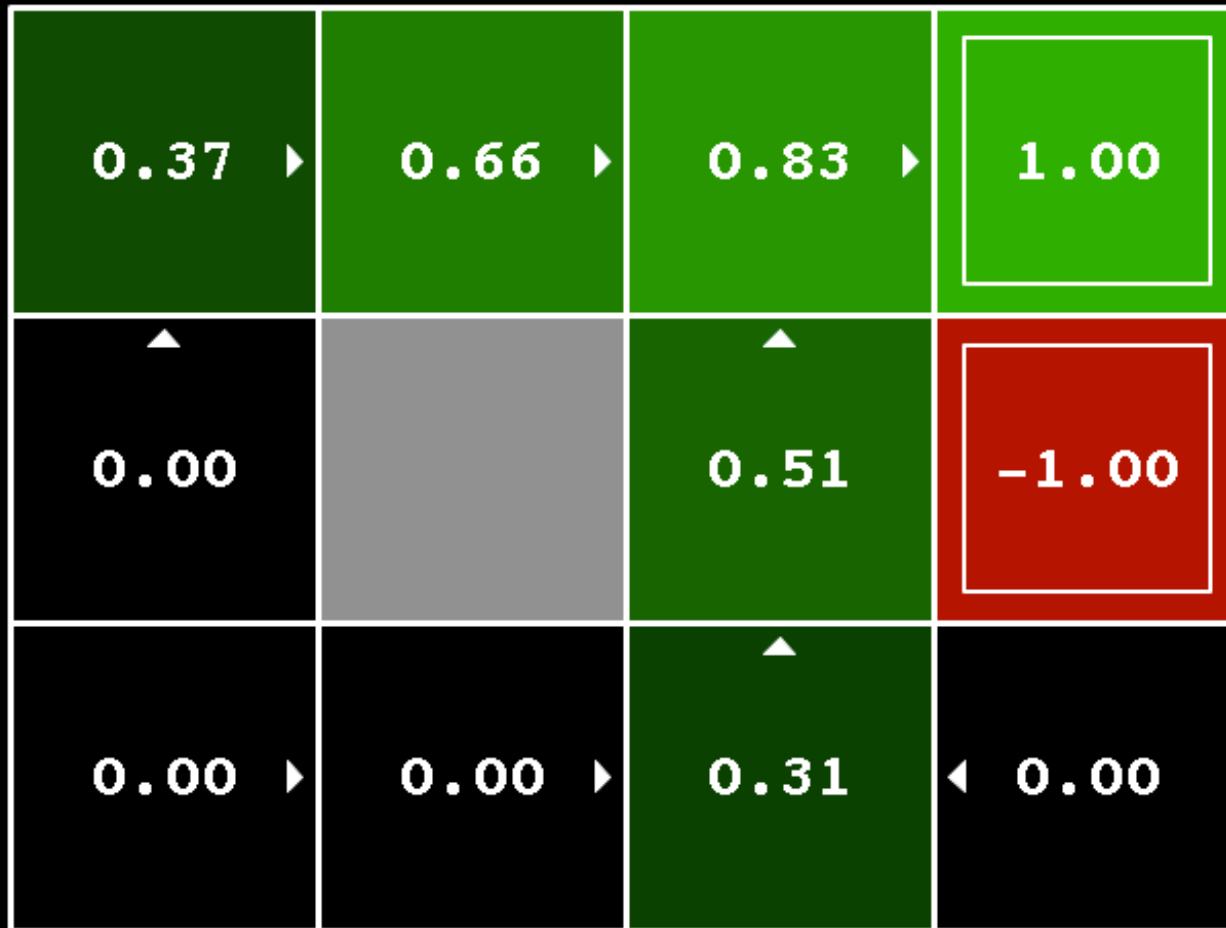
noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$



**VALUES AFTER 3 ITERATIONS**

# Value Iteration in Gridworld

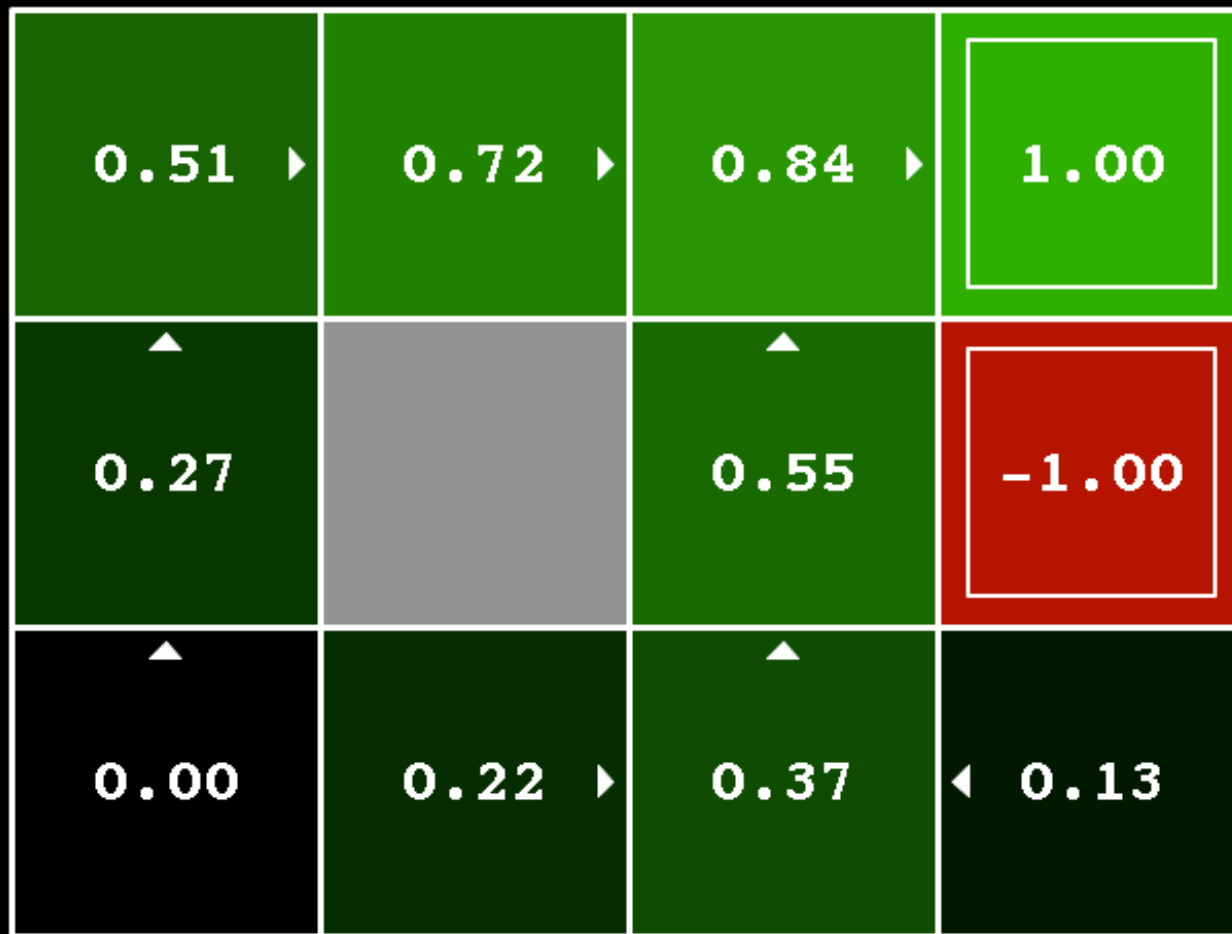
noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$



**VALUES AFTER 4 ITERATIONS**

# Value Iteration in Gridworld

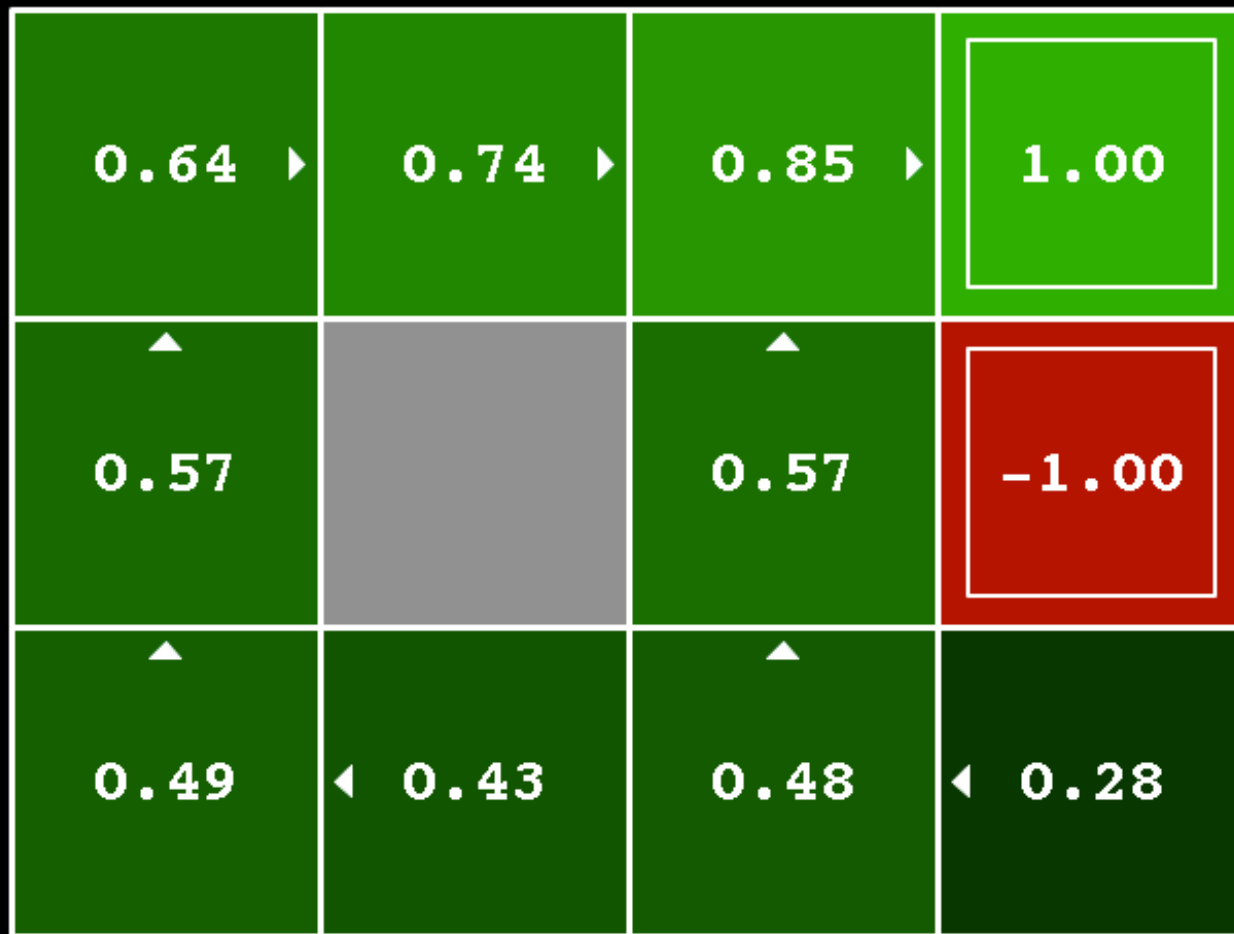
noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$



VALUES AFTER 5 ITERATIONS

# Value Iteration in Gridworld

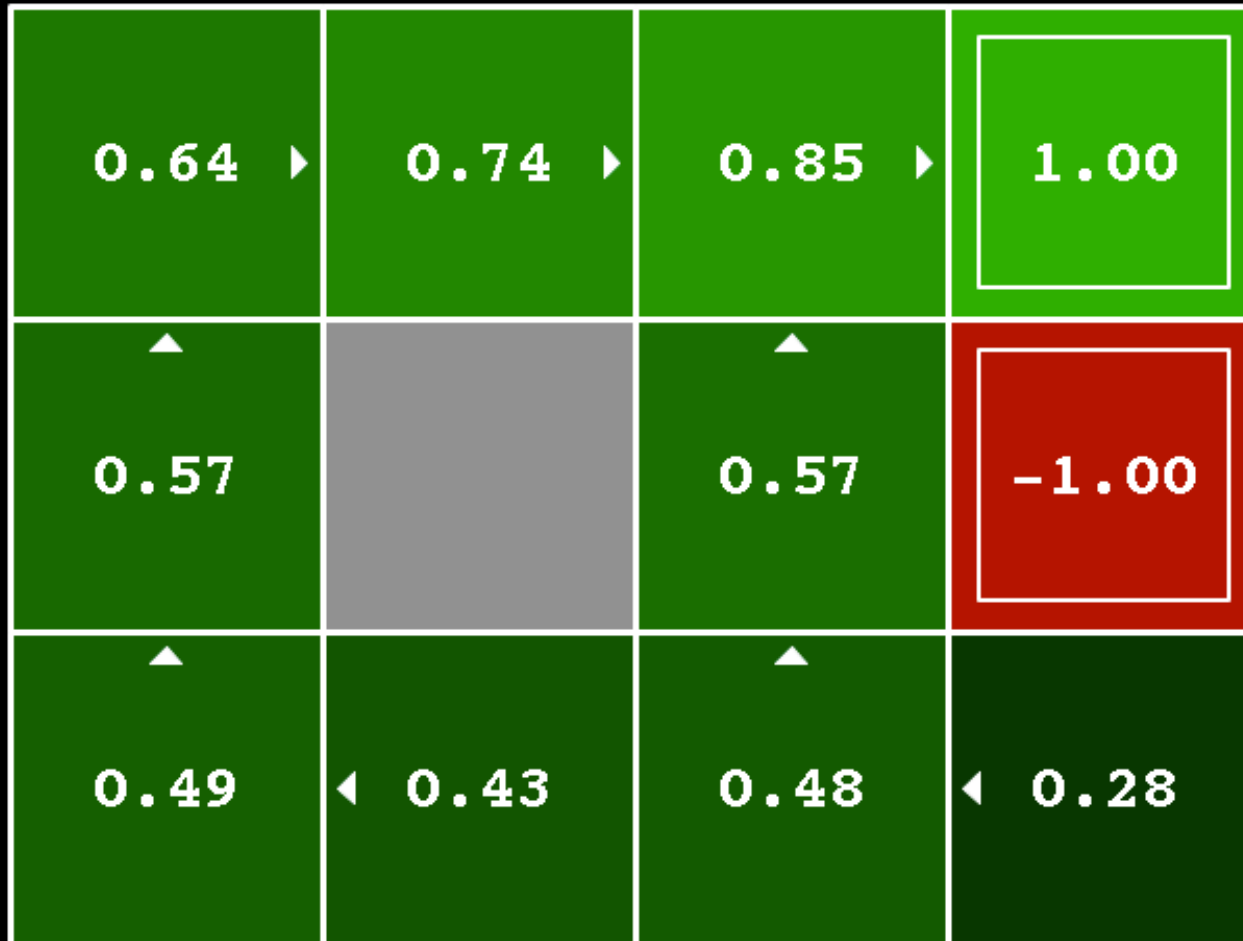
noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$



VALUES AFTER 100 ITERATIONS

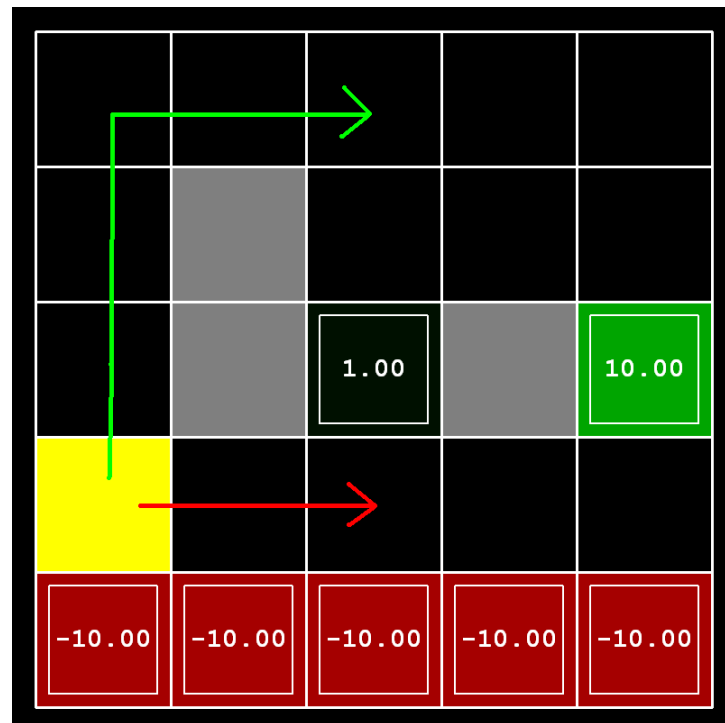
# Value Iteration in Gridworld

noise = 0.2,  $\gamma = 0.9$ , two terminal states with  $R = +1$  and  $-1$



**VALUES AFTER 1000 ITERATIONS**

# Exercise 1: Effect of discount, noise



- (a) Prefer the close exit (+1), risking the cliff (-10)
- (b) Prefer the close exit (+1), but avoiding the cliff (-10)
- (c) Prefer the distant exit (+10), risking the cliff (-10)
- (d) Prefer the distant exit (+10), avoiding the cliff (-10)

- (1)  $\gamma = 0.1$ , noise = 0.5
- (2)  $\gamma = 0.99$ , noise = 0
- (3)  $\gamma = 0.99$ , noise = 0.5
- (4)  $\gamma = 0.1$ , noise = 0



# Exercise 1 Solution

|           |        |           |        |           |
|-----------|--------|-----------|--------|-----------|
| 0.00 ▸    | 0.00 ▸ | 0.01<br>▼ | 0.01 ▸ | 0.10<br>▼ |
| 0.00<br>▼ |        | 0.10<br>▼ | 0.10 ▸ | 1.00<br>▼ |
| 0.00<br>▼ |        | 1.00      |        | 10.00     |
| 0.00 ▸    | 0.01 ▸ | 0.10<br>▲ | 0.10 ▸ | 1.00<br>▲ |
| -10.00    | -10.00 | -10.00    | -10.00 | -10.00    |

(a) Prefer close exit (+1), risking the cliff (-10) ---  $\gamma = 0.1$ , noise = 0

# Exercise 1 Solution

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 0.00 ▶    | 0.00 ▶    | 0.00<br>▼ | 0.00<br>▼ | 0.03<br>▼ |
| ▲<br>0.00 |           | 0.05<br>▼ | 0.03 ▶    | 0.51<br>▼ |
| 0.00<br>▼ |           | 1.00      |           | 10.00     |
| ▲<br>0.00 | ▲<br>0.00 | ▲<br>0.05 | ▲<br>0.01 | ▲<br>0.51 |
| -10.00    | -10.00    | -10.00    | -10.00    | -10.00    |

(b) Prefer close exit (+1), avoiding the cliff (-10) --  $\gamma = 0.1$ , noise = 0.5

# Exercise 1 Solution

|           |        |        |        |           |
|-----------|--------|--------|--------|-----------|
| 9.41 ▶    | 9.51 ▶ | 9.61 ▶ | 9.70 ▶ | 9.80<br>▼ |
| 9.32<br>▼ |        | 9.70 ▶ | 9.80 ▶ | 9.90<br>▼ |
| 9.41<br>▼ |        | 1.00   |        | 10.00     |
| 9.51 ▶    | 9.61 ▶ | 9.70 ▶ | 9.80 ▶ | 9.90<br>▲ |
| -10.00    | -10.00 | -10.00 | -10.00 | -10.00    |

(c) Prefer distant exit (+1), risking the cliff (-10) --  $\gamma = 0.99$ , noise = 0

# Exercise 1 Solution



(d) Prefer distant exit (+1), avoid the cliff (-10) --  $\gamma = 0.99$ , noise = 0.5

# Value Iteration Convergence

**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

$$\forall S \in \mathcal{S} : \quad V^*(s) = \max_A \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

- Now we know how to act for infinite horizon with discounted rewards!

- Run value iteration till convergence.
- This produces  $V^*$ , which in turn tells us how to act, namely following:

$$\pi^*(s) = \arg \max_{a \in A} \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

- Note: the infinite horizon optimal policy is stationary, i.e., the optimal action at a state  $s$  is the same action at all times. (Efficient to store!)

# Convergence and Contractions

- Define the max-norm:  $\|U\| = \max_s |U(s)|$

- Theorem: For any two approximations U and V

$$\|U_{i+1} - V_{i+1}\| \leq \gamma \|U_i - V_i\|$$

- 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:

$$\|V_{i+1} - V_i\| < \epsilon, \Rightarrow \|V_{i+1} - V^*\| < 2\epsilon\gamma/(1 - \gamma)$$

- I.e. once the change in our approximation is small, it must also be close to correct

# Outline

- Optimal Control

=

given an MDP ( $S, A, T, R, \gamma, H$ )

find the optimal policy  $\pi^*$

- Exact Methods:



Value Iteration

- ***Policy Iteration***

- Linear Programming

For now: discrete state-action spaces as they are simpler to get the main concepts across. Will consider continuous spaces later!

# Policy Evaluation

- Recall value iteration iterates:

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

- Policy evaluation:

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

- At convergence:

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



# Exercise 2

Consider a stochastic policy  $\mu(a|s)$ , where  $\mu(a|s)$  is the probability of taking action  $a$  when in state  $s$ . Which of the following is the correct value iteration update to perform policy evaluation for this stochastic policy?

1.  $V_{i+1}^\mu(s) \leftarrow \max_a \sum_{s'} T(s, a, s') (R(s, a, s') + \gamma V_i^\mu(s'))$
2.  $V_{i+1}^\mu(s) \leftarrow \sum_{s'} \sum_a \mu(a|s) T(s, a, s') (R(s, a, s') + \gamma V_i^\mu(s'))$
3.  $V_{i+1}^\mu(s) \leftarrow \sum_a \mu(a|s) \max_{s'} T(s, a, s') (R(s, a, s') + \gamma V_i^\mu(s'))$

# Policy Iteration

- Alternative approach:
  - **Step 1: Policy evaluation:** calculate utilities for some fixed policy (not optimal utilities!) until convergence
  - **Step 2: Policy improvement:** update policy using one-step look-ahead with resulting converged (but not optimal!) utilities as future values
  - Repeat steps until policy converges
- This is **policy iteration**
  - It's still optimal!
  - Can converge faster under some conditions

# Policy Evaluation Revisited

- Idea 1: 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 2: it's just a linear system, solve with Matlab (or whatever),  
variables:  $V^\pi(s)$ ,  
constants:  $T, R$

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

# Policy Iteration Guarantees

Policy Iteration iterates over:

- Policy evaluation: with fixed current policy  $\pi$ , find values with simplified Bellman updates:

- Iterate until values converge

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

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

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

**Theorem.** Policy iteration is guaranteed to converge and at convergence, the current policy and its value function are the optimal policy and the optimal value function!

Proof sketch:

- (1) *Guarantee to converge:* In every step the policy improves. This means that a given policy can be encountered at most once. This means that after we have iterated as many times as there are different policies, i.e.,  $(\text{number actions})^{(\text{number states})}$ , we must be done and hence have converged.
- (2) *Optimal at convergence:* by definition of convergence, at convergence  $\pi_{k+1}(s) = \pi_k(s)$  for all states  $s$ . This means  $\forall s \ V^{\pi_k}(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_k}(s')]$   
Hence  $V^{\pi_k}$  satisfies the Bellman equation, which means  $V^{\pi_k}$  is equal to the optimal value function  $V^*$ .

# Outline

- Optimal Control

=

given an MDP ( $S, A, T, R, \gamma, H$ )

find the optimal policy  $\pi^*$

- Exact Methods:



Value Iteration



Policy Iteration

- ***Linear Programming***

For now: discrete state-action spaces as they are simpler to get the main concepts across. Will consider continuous spaces later!

# Infinite Horizon Linear Program

- Recall, at value iteration convergence we have

$$\forall s \in S : \quad V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

- LP formulation to find  $V^*$ :

$$\begin{aligned} \min_V \quad & \sum_s \mu_0(s) V(s) \\ \text{s.t.} \quad & \forall s \in S, \forall a \in A : \\ & V(s) \geq \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')] \end{aligned}$$

$\mu_0$  is a probability distribution over  $S$ , with  $\mu_0(s) > 0$  for all  $s \in S$ .

**Theorem.**  $V^*$  is the solution to the above LP.

# Theorem Proof

Let  $F$  be the Bellman operator, i.e.,  $V_{i+1}^* = F(V_i)$ . Then the LP can be written as:

$$\begin{array}{ll} \min_V & \mu_0^\top V \\ \text{s.t.} & V \geq F(V) \end{array}$$

**Property: Monotonicity.** If  $U \geq V$  then  $F(U) \geq F(V)$ .

Hence, if  $V \geq F(V)$  then  $F(V) \geq F(F(V))$ , and by repeated application,  $V \geq F(V) \geq F^2V \geq F^3V \geq \dots \geq F^\infty V = V^*$ .

Any feasible solution to the LP must satisfy  $V \geq F(V)$ , and hence must satisfy  $V \geq V^*$ . Hence, assuming all entries in  $\mu_0$  are positive,  $V^*$  is the optimal solution to the LP.

# Dual Linear Program

$$\begin{aligned} \max_{\lambda} \quad & \sum_{s \in S} \sum_{a \in A} \sum_{s' \in S} \lambda(s, a) T(s, a, s') R(s, a, s') \\ \text{s.t.} \quad & \forall s' \in S : \sum_{a' \in A} \lambda(s', a') = \mu_0(s) + \gamma \sum_{s \in S} \sum_{a \in A} \lambda(s, a) T(s, a, s') \end{aligned}$$

- Interpretation:

- $\lambda(s, a) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s, a_t = a)$

- Equation 2: ensures  $\lambda$  has the above meaning

- Equation 1: maximize expected discounted sum of rewards

- Optimal policy:  $\pi^*(s) = \arg \max_a \lambda(s, a)$



# Outline

- Optimal Control

=

given an MDP ( $S, A, T, R, \gamma, H$ )

find the optimal policy  $\pi^*$

- Exact Methods:



Value Iteration



Policy Iteration



Linear Programming

For now: discrete state-action spaces as they are simpler to get the main concepts across. Will consider continuous spaces later!

# Today and forthcoming lectures

- Optimal control: provides general computational approach to tackle control problems.
  - Dynamic programming / Value iteration
    - ✓ Exact methods on discrete state spaces (DONE!)
      - Discretization of continuous state spaces
      - Function approximation
      - Linear systems
      - LQR
      - Extensions to nonlinear settings:
        - Local linearization
        - Differential dynamic programming
  - Optimal Control through Nonlinear Optimization
    - Open-loop
    - Model Predictive Control
  - Examples:

