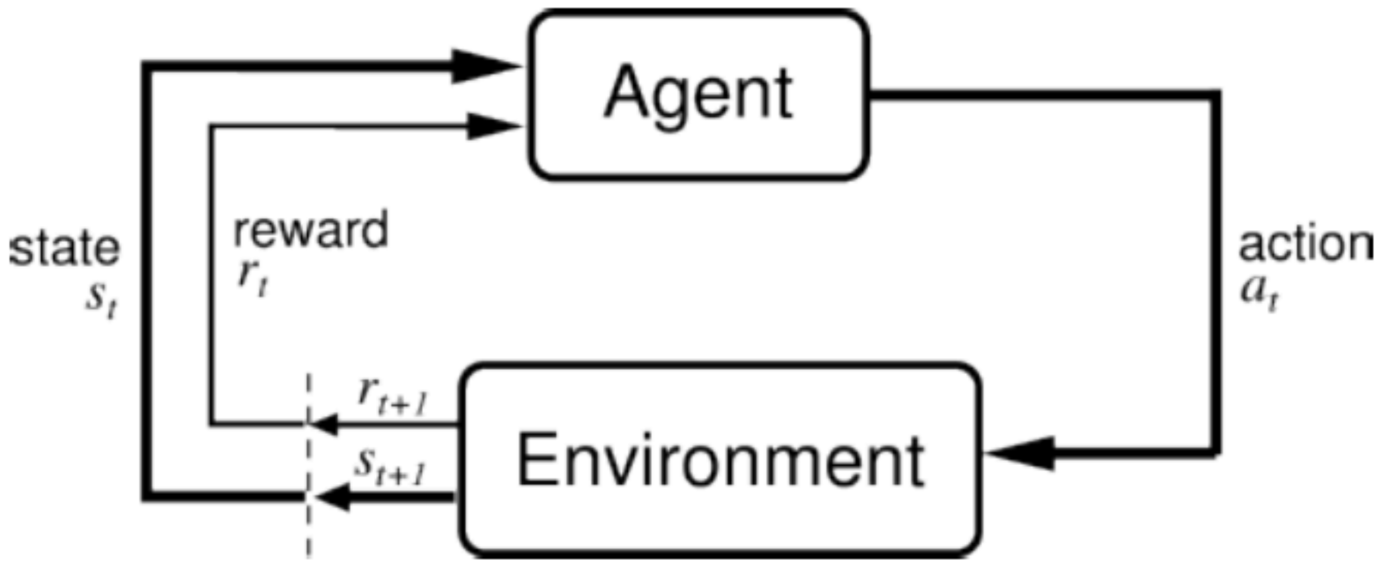
Markov Decision Problems

D.A. Forsyth, UIUC

Topics

- Vocabulary
- Simplest imitation learning and DAGGER
  - to set up possible projects, and answer Q1, Q2
- Simple reinforcement learning ideas
- More imitation learning; inverse reinforcement learning
  - and its variants and problems

Markov Decision Process



Assumption: agent gets to observe the state

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

Abbeel slides

Model

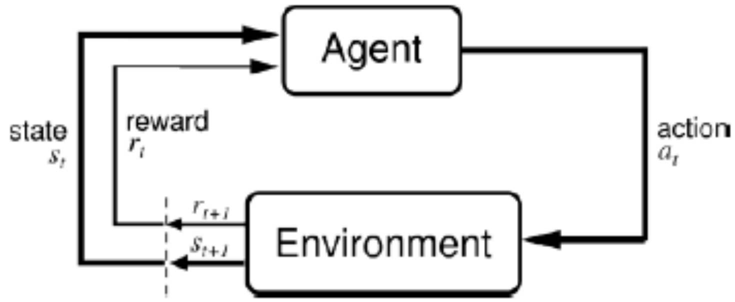
- At time 0, environment samples initial state
  - agent is in that state
- Then for t=0 till done
  - agent chooses action
  - environment samples new state conditioned on action, old state
  - environment samples reward conditioned on action, old state, new state
  - agent gets that reward and moves into new state
- Policy
  - what action to take in each state
    - this could be stochastic
- Maximise total discounted reward

Examples

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

Abbeel slides

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



Given

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

Goal:

- Find π : S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

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

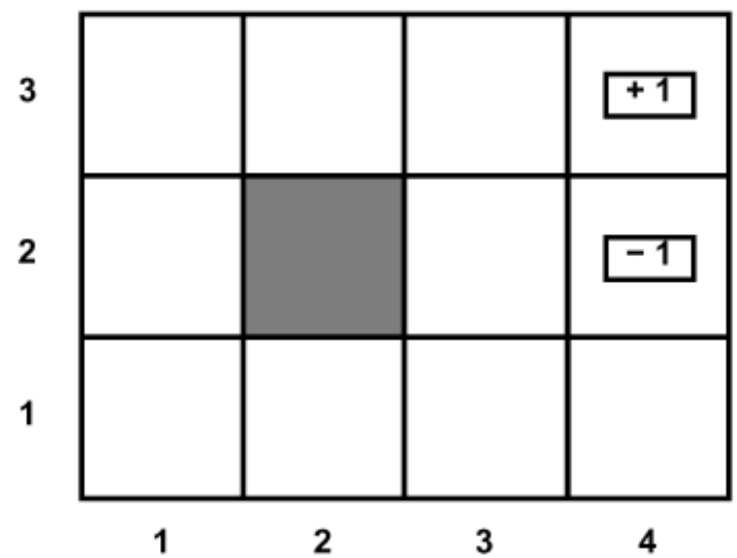
This is usually discounted by gamma

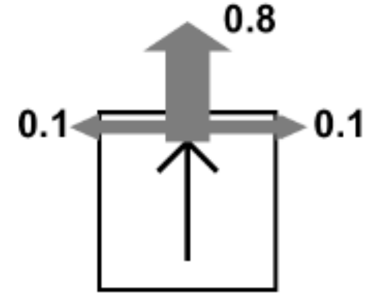
Abbeel slides

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

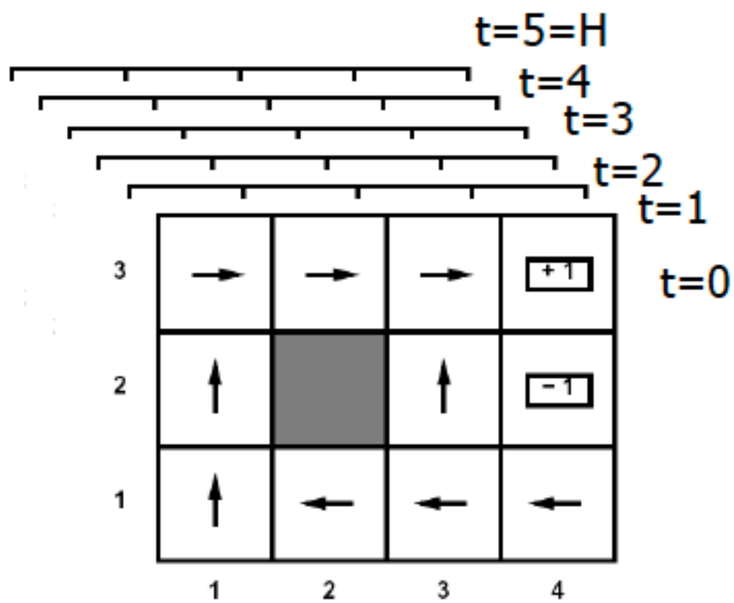
And this is true for the other three; 80% of the time you go where you intended, 10% at right angles one way 10% the other





Solving MDPs

- In an MDP, we want an optimal policy π^*: S x 0:H → A
  - A policy π 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, γ, H)

  find the optimal policy π^*

- Exact Methods:
  - ***Value Iteration***
  - Policy Iteration

Value iteration

- Idea:
  - value of a state=expected reward of proceeding optimally from that state
  - if we knew the value of each state, choosing an action is easy
    - take the one with the best expected yield
  - cf HMM inference reasoning
- Idea:
  - we could estimate the value of a state
    - set the value of every state to something
    - now for a given state, compute the expected value of best action
      - replace value with that and continue

Value Iteration

- Algorithm:
  - Start with $V_0^*(s) = 0$ for all s.
  - For i=1, … , H

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

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

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

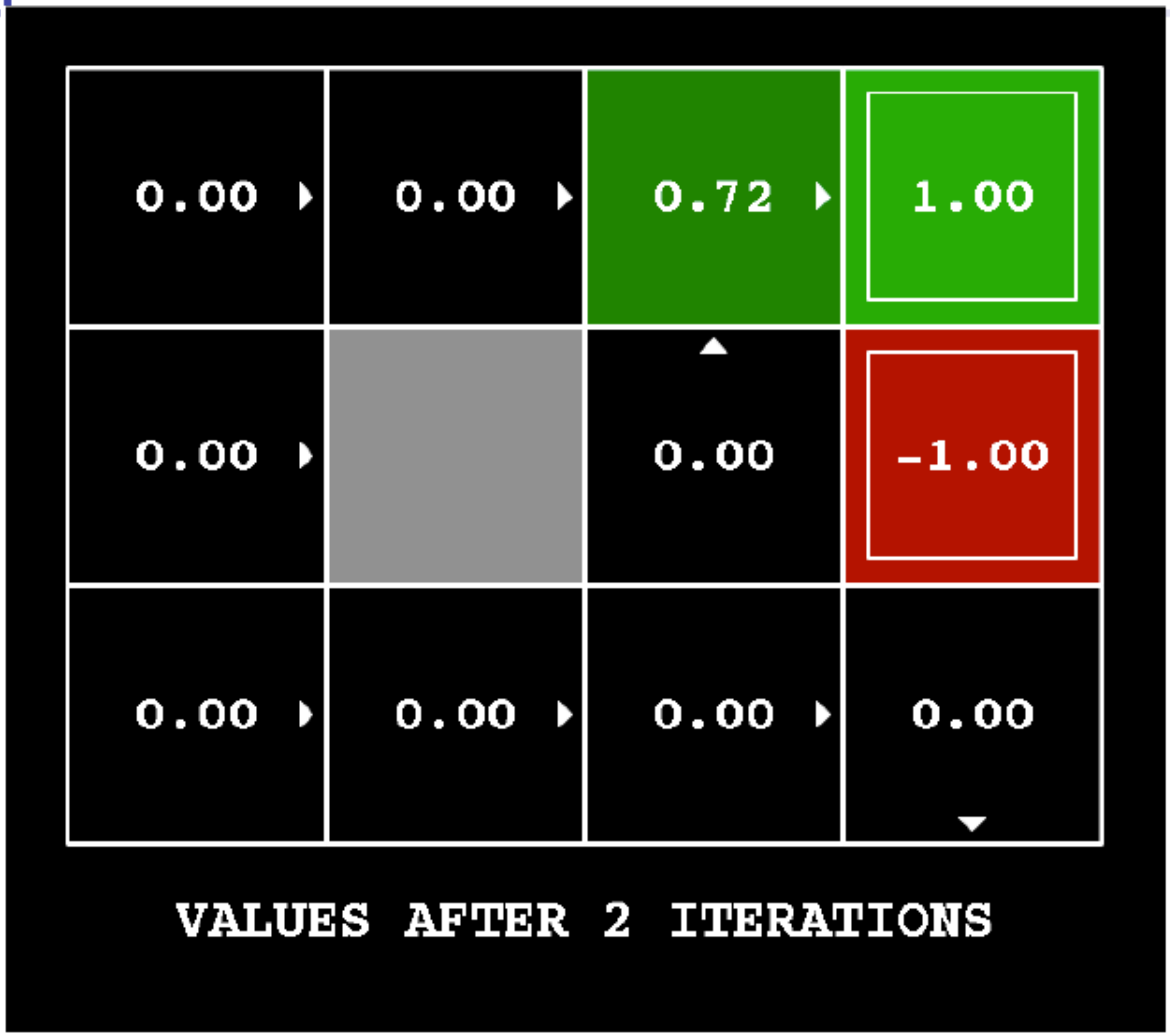
- $V_i^*(s)$ = the expected sum of rewards accumulated when starting from state s and acting optimally for a horizon of i steps

Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

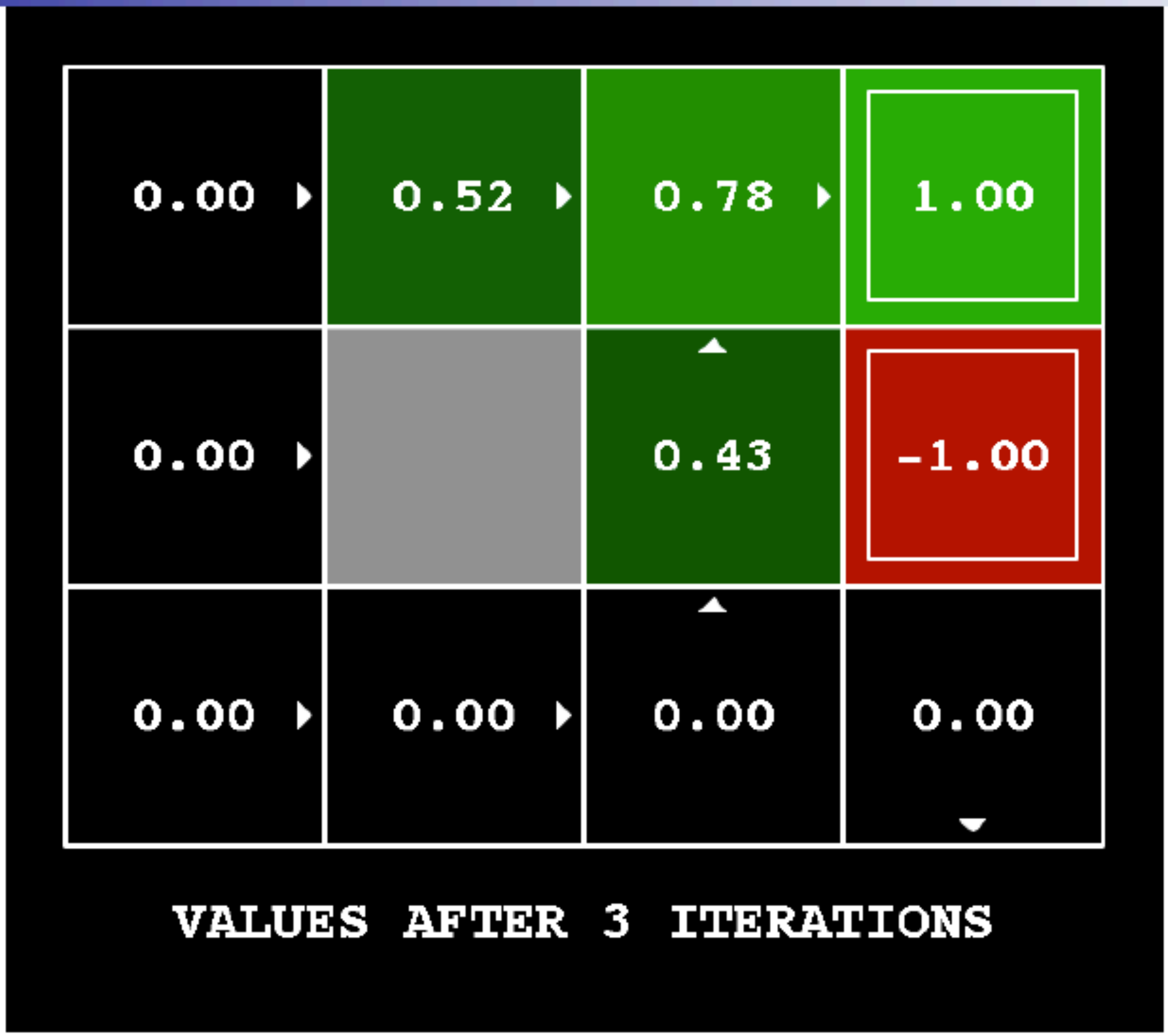
Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1



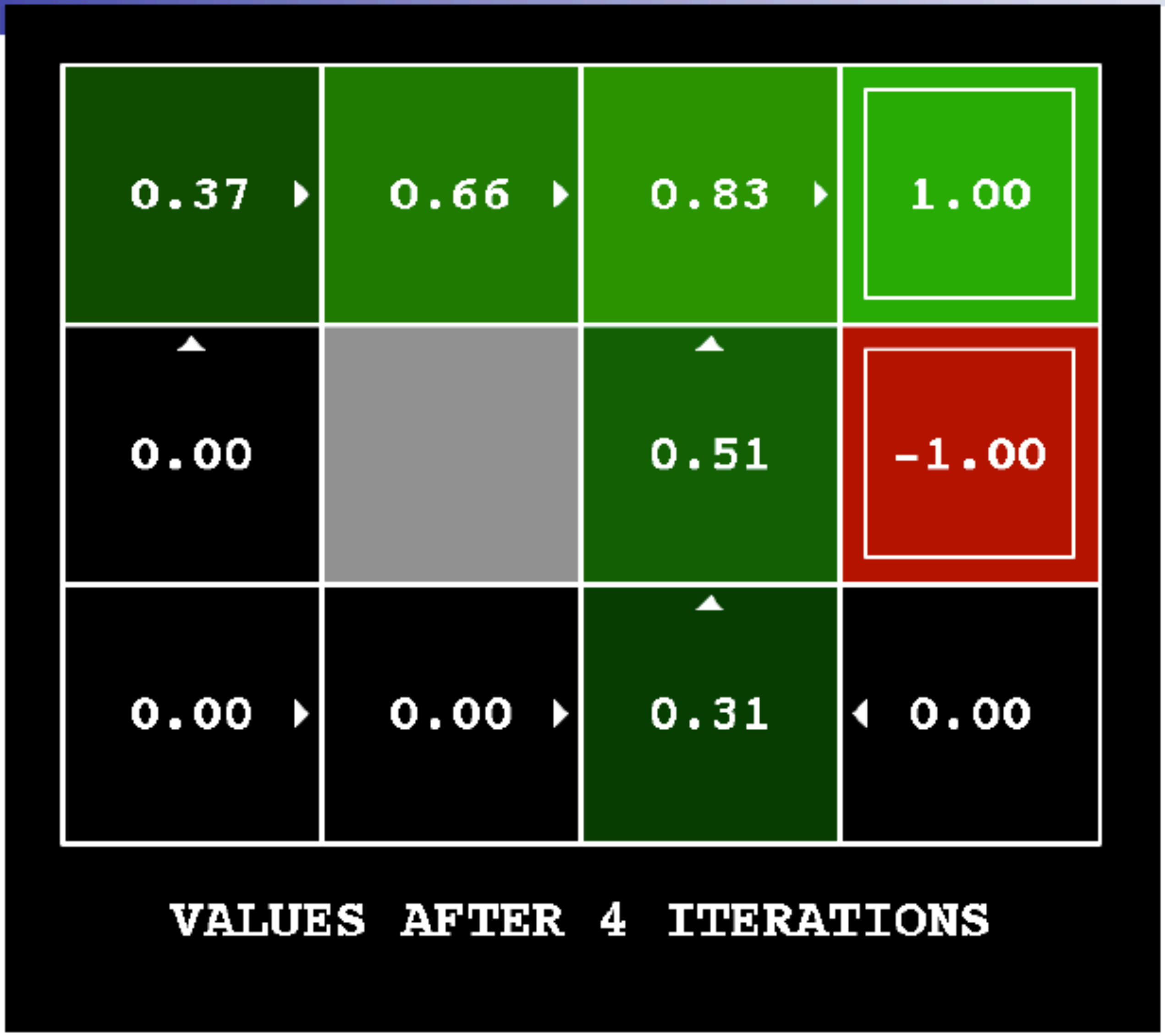
Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1



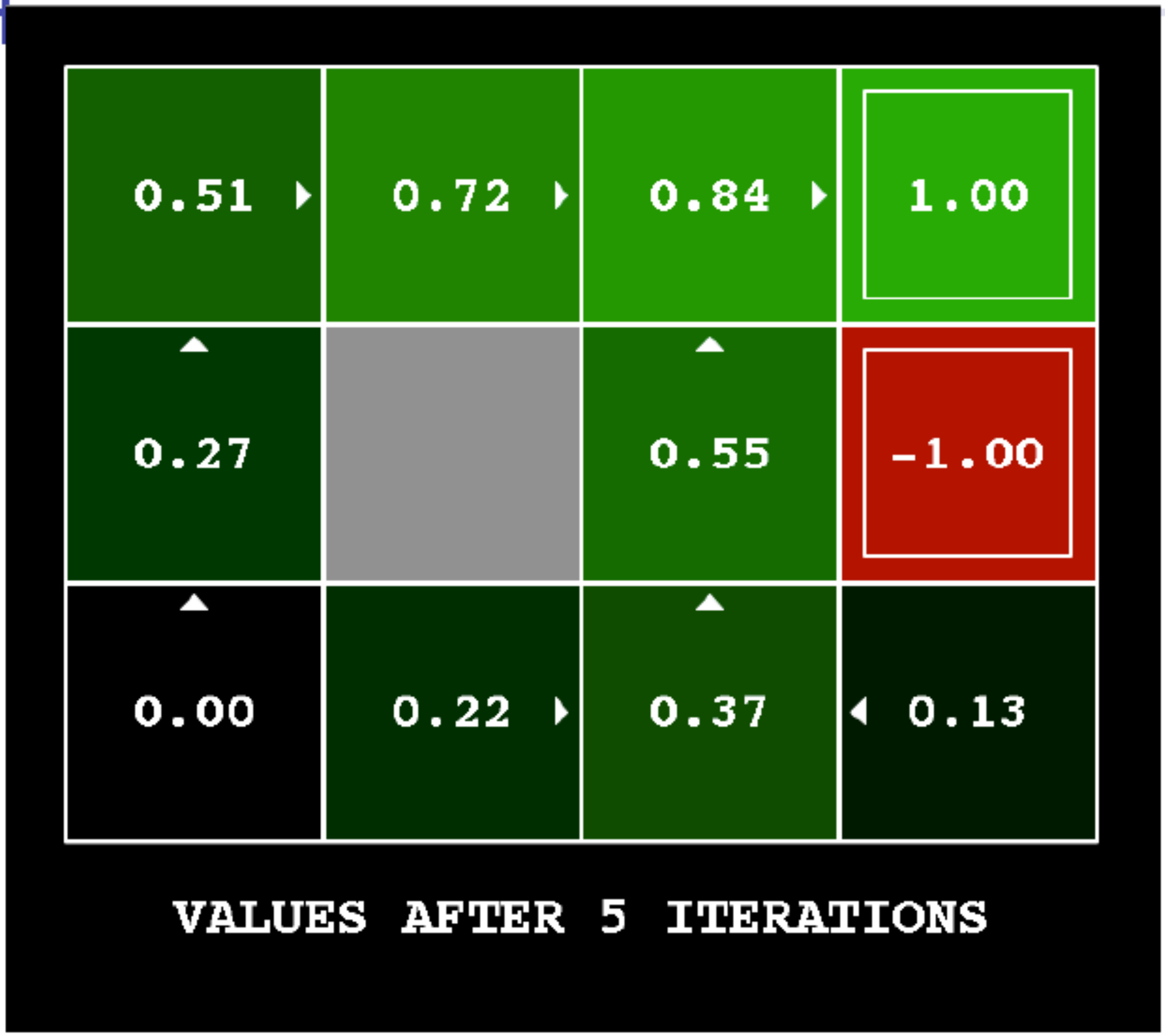
Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1



Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

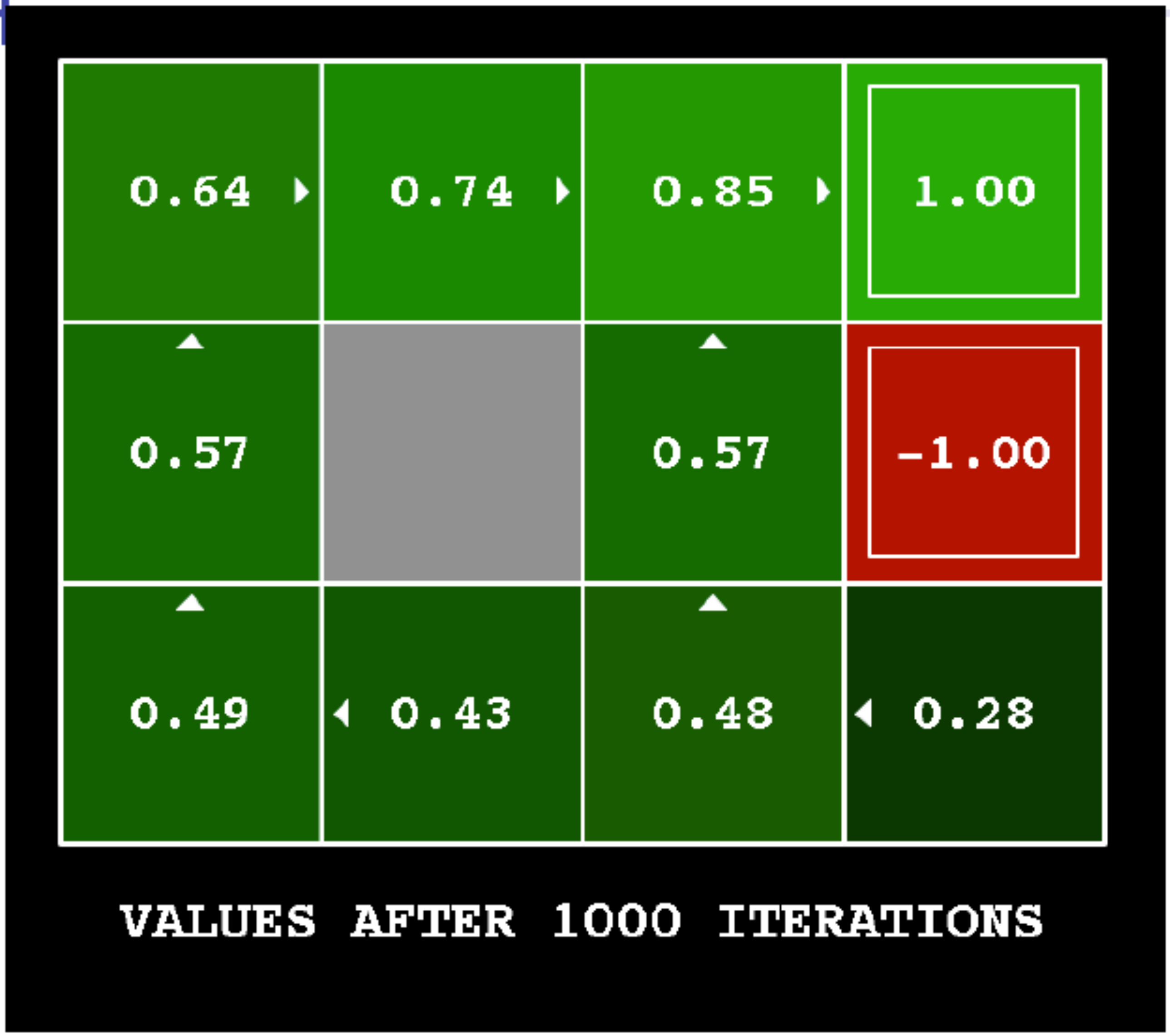


Value Iteration in Gridworld

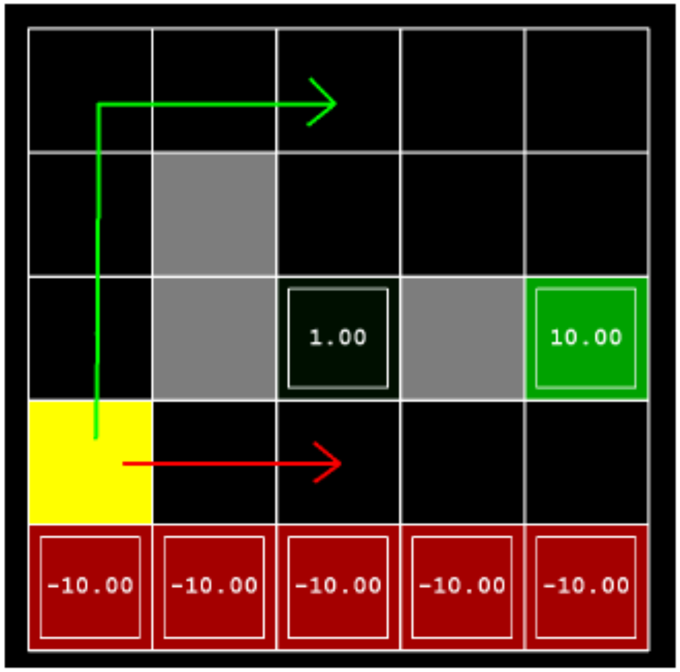
noise = 0.2, γ =0.9, two terminal states with R = +1 and -1

Value Iteration in Gridworld

noise = 0.2, γ =0.9, two terminal states with R = +1 and -1



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)

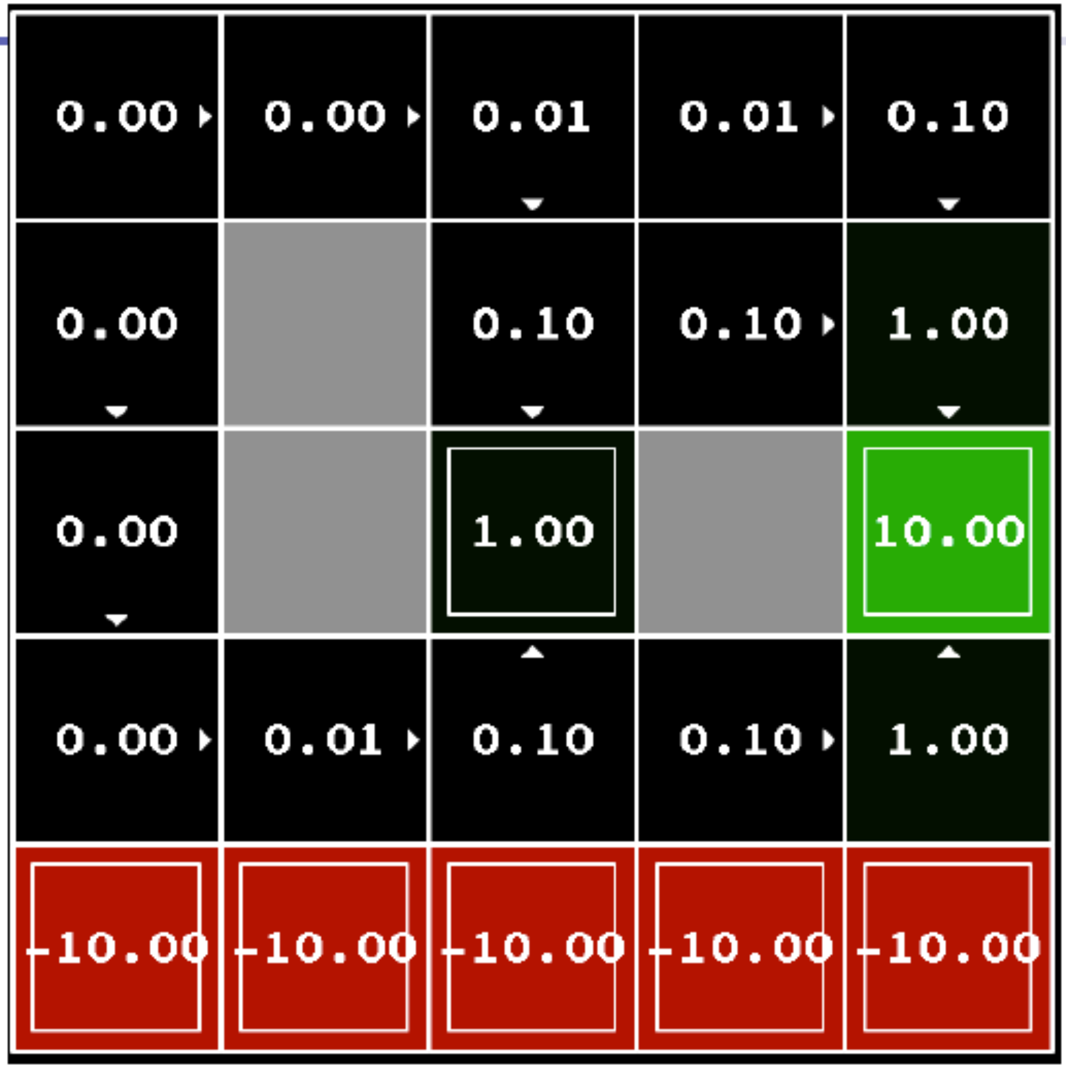
(1) γ = 0.1, noise = 0.5

(2) γ = 0.99, noise = 0

(3) γ = 0.99, noise = 0.5

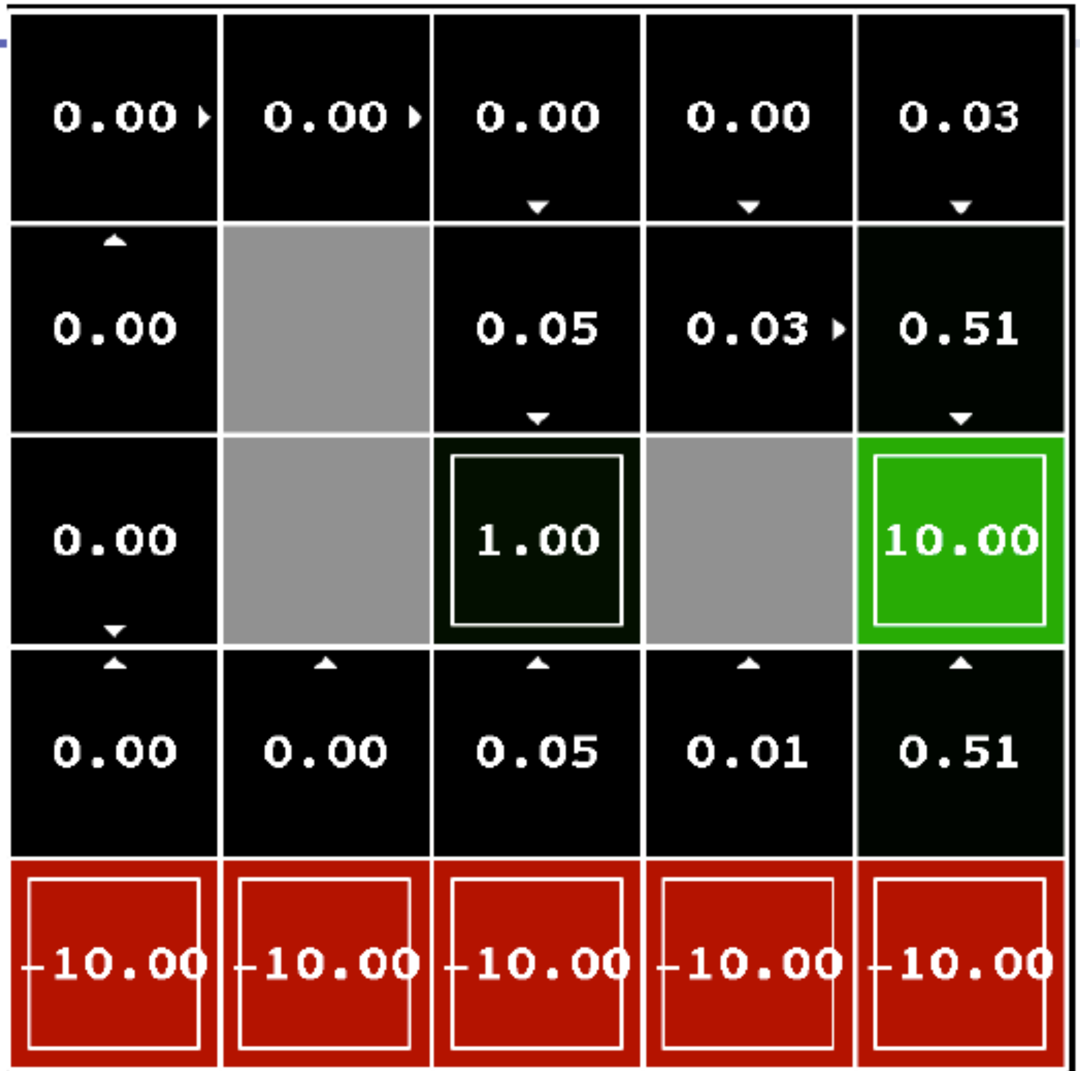
(4) γ = 0.1, noise = 0

Exercise 1 Solution



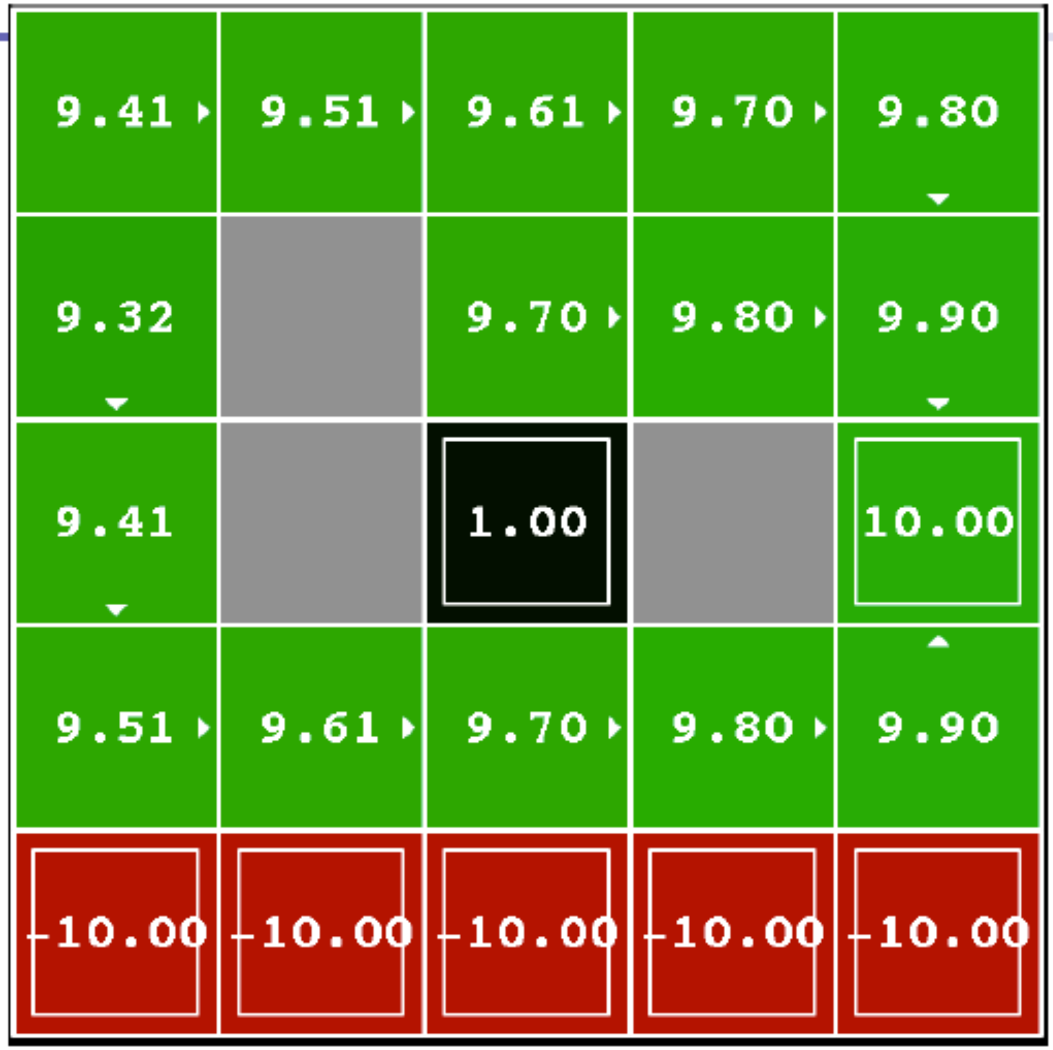
(a) Prefer close exit (+1), risking the cliff (-10) --- γ = 0.1, noise = 0

Exercise 1 Solution



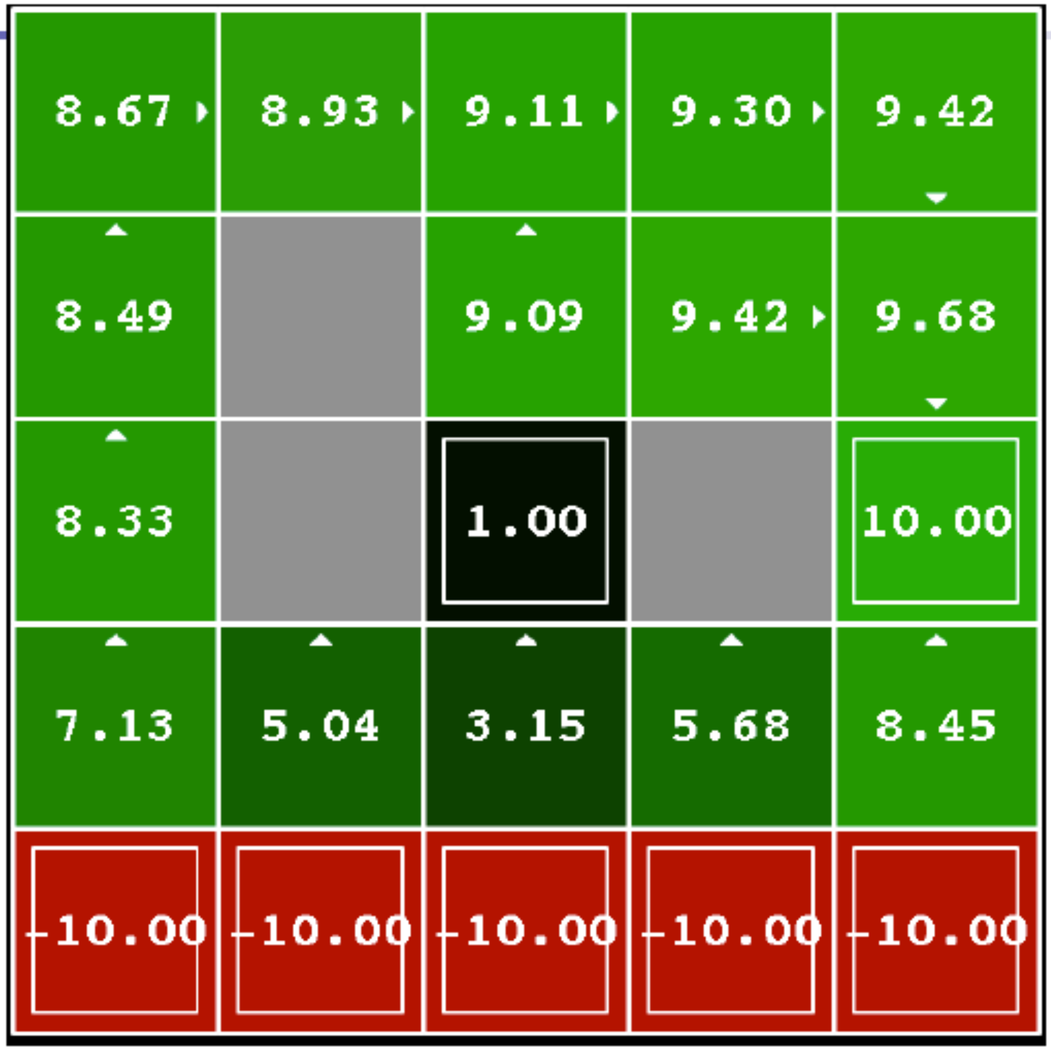
(b) Prefer close exit (+1), avoiding the cliff (-10) -- $\gamma = 0.1$, noise = 0.5

Exercise 1 Solution



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

Exercise 1 Solution



(d) Prefer distant exit (+1), avoid the cliff (-10) -- γ = 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

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

- 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!)

But it’s not really all over…

- What if:
  - there are lots of states?
  - we don’t know T?
  - we don’t know R?

Policy iteration

- Idea:
    - evaluate some policy
    - then make it better

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 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

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

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

Hence V^{π_k} satisfies the Bellman equation, which means V^{π_k} is equal to the optimal value function V*.