AA203 Optimal and Learning-based Control

Lecture 6

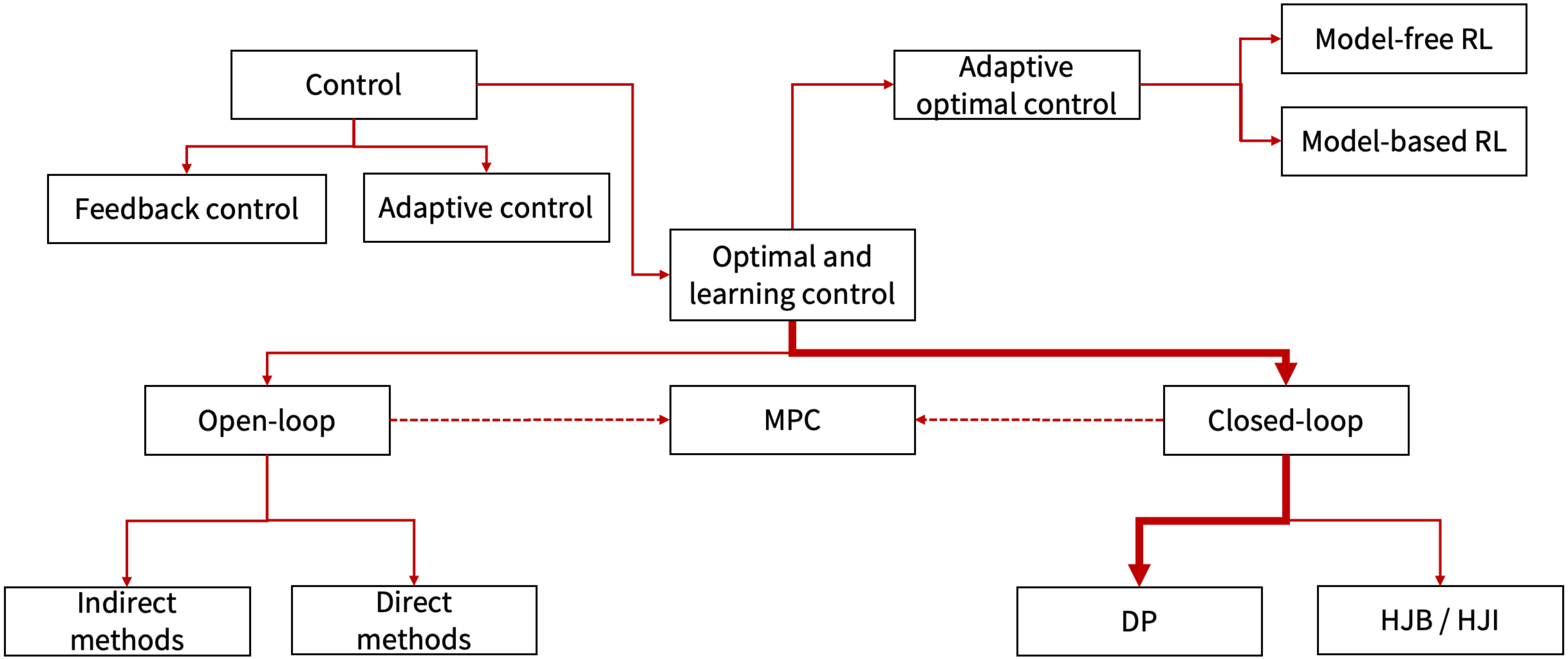
Stochastic Dynamic Programming

Autonomous Systems Laboratory
Daniele Gammelli

Stanford University

Autonomous Systems Laboratory
Stanford Aeronautics & Astronautics

Roadmap



Control
Feedback control
Adaptive control
Optimal and learning control
Adaptive optimal control
Model-free RL
Model-based RL
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI

Outline

Stochastic Optimal Control: Markov Decision Process (MDP)

The dynamic programming algorithm (stochastic case)

Stochastic LQR

Infinite-Horizon MDPs:

- Exact Methods:
  - (Policy Evaluation)
  - Value Iteration
  - Policy Iteration

Stochastic Optimal Control Problem: Markov Decision Problem (MDP)

- **System**: $\boldsymbol{x}_{k+1} = f_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right), k = 0,\ldots,N-1$
- **Probability distribution**: $\boldsymbol{w}_k \sim P_k\left(\cdot \mid \boldsymbol{x}_k, \boldsymbol{u}_k\right)$
- **Control constraints:** $\boldsymbol{u}_k \in U\left(\boldsymbol{x}_k\right)$
- **Policies:** $\pi = \left\{\pi_0\ldots,\pi_{N-1}\right\}$, where $\boldsymbol{u}_k = \pi_k\left(\boldsymbol{x}_k\right)$
- **Expected Cost**:

$$J_\pi\left(\boldsymbol{x}_0\right) = \mathbb{E}_{\boldsymbol{w}_k, k=0,\ldots,N-1}\left[g_N\left(\boldsymbol{x}_N\right) + \sum_{k=0}^{N-1} g_k\left(\boldsymbol{x}_k, \pi_k\left(\boldsymbol{x}_k\right), \boldsymbol{w}_k\right)\right]$$

Stochastic Optimal Control Problem:

$$J^*\left(x_0\right) = \min_\pi J_\pi\left(x_0\right)$$

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal **closed-loop** policy
- Additive cost (central assumption in DP)
- Risk-neutral formulation

Other communities use different notation:
[Powell, W. B. *AI, OR and control theory: A Rosetta Stone for stochastic optimization.* Princeton University, 2012.]

Principle of optimality (stochastic case)

Principle of optimality:

- Let $\pi^* := \left\{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\right\}$ be an optimal policy
- Consider the tail subproblem

$$\mathbb{E}_{w_k}\left[g_N\left(\boldsymbol{x}_N\right) + \sum_{k=i}^{N-1} g_k\left(\boldsymbol{x}_k, \pi_k\left(\boldsymbol{x}_k\right), \boldsymbol{w}_k\right)\right]$$

the tail policy $\left\{\pi_i^*, \ldots, \pi_{N-1}^*\right\}$ is optimal for the tail subproblem

Intuition:

- DP first solves ALL tail subproblems at the final stage
- At the generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length

DP Algorithm (stochastic case)

Like in the deterministic case, start with:

$$J_N^*\left(x_N\right) = g_N\left(x_N\right)$$

and iterate backwards in time using

$$J_k^*\left(\boldsymbol{x}_k\right) = \min_{\boldsymbol{u}_k \in U\left(\boldsymbol{x}_k\right)} \mathbb{E}_{w_k}\left[g_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right) + J_{k+1}^*\left(f\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right)\right)\right], \quad k = 0, \ldots, N-1$$

for which the optimal cost $J^*(\mathbf{x}_0)$ is equal to $J_0(\mathbf{x}_0)$ and the optimal policy is constructed by setting

$$\pi_k^*\left(\boldsymbol{x}_k\right) = \operatorname*{argmin}_{\boldsymbol{u}_k \in U\left(\boldsymbol{x}_k\right)} \mathbb{E}_{w_k}\left[g_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right) + J_{k+1}^*\left(f\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k\right)\right)\right]$$

Example: Inventory Control Problem

$x_k \in \mathbb{N}$: stock available
$u_k \in \mathbb{N}$: inventory
$w_k \in \mathbb{N}$: demand

Dynamics: $x_{k+1} = \max\left(0, x_k + u_k - w_k\right)$

Constraints: $x_k + u_k \leq 2$

Probabilistic structure: $p(w_k = 0) = 0.1$
$p(w_k = 1) = 0.7$
$p(w_k = 2) = 0.2$

Objective:

$$\mathbb{E}_{w_k}\left[\underbrace{0}_{g_3(x_3)} + \sum_{k=0}^{2} \underbrace{\left(u_k + \left(x_k + u_k - w_k\right)^2\right)}_{g_k(x_k, u_k, w_k)} \right]$$

More generally, could imagine costs:
$H(x_k)$: holding inventory
$B(u_k)$: buying inventory
$S(x_k, u_k, w_k)$: selling (matching stock with demand)

Example: Inventory Control Problem

Algorithm takes the form

$$J_k^*\left(x_k\right) = \min_{0\le u_k\le 2-x_k} \mathbb{E}_{w_k}\left[u_k + \left(x_k + u_k - w_k\right)^2 + J_{k+1}^*\left(\max\left(0, x_k + u_k - w_k\right)\right)\right]$$

for $k = 0,1,2$

For example

$$J_2^*(0) = \min_{u_2=0,1,2} \mathbb{E}_{w_2}\left[u_2 + \left(u_2 - w_2\right)^2\right] =$$

$$\min_{u_2=0,1,2} u_2 + 0.1\left(u_2\right)^2 + 0.7\left(u_2 - 1\right)^2 + 0.2\left(u_2 - 2\right)^2$$

Which yields $J_2^*(0) = 1.3$ and $\pi_2^*(0) = 1$

Example: Inventory Control Problem

Final solution:

$$J_0^*(0) = 3.7$$

$$J_0^*(1) = 2.7$$

$$J_0^*(2) = 2.818$$

(See this spreadsheet)

Stochastic LQR

Find control policy that minimizes

$$\mathbb{E}_{w_k}\left[\frac{1}{2}\boldsymbol{x}_N^T Q \boldsymbol{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^T R_k \boldsymbol{u}_k\right)\right]$$

Subject to

- Dynamics $\mathbf{x}_{k+1} = A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k, \quad k \in \{0,1,\ldots,N-1\}$

with $\boldsymbol{x}_0 \sim \mathcal{N}\left(\overline{\boldsymbol{x}_0}, \Sigma_{\boldsymbol{x}_0}\right), \left\{\boldsymbol{w}_k \sim \mathcal{N}\left(\mathbf{0}, \Sigma_{\boldsymbol{w}_k}\right)\right\}$ independent and Gaussian vectors

Stochastic LQR

As in the deterministic case, with $J^*_{k+1}\left(\mathbf{x}_{k+1}\right) = \frac{1}{2}\mathbf{x}_{k+1}^T P_{k+1}\mathbf{x}_{k+1}$

$$
\begin{aligned}
J^*_k\left(\mathbf{x}_{k+1}\right) &= \min_{\mathbf{u}_k} \mathbb{E}_{\mathbf{w}_k}\left[g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right) + J^*_{k+1}\left(f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)\right)\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\mathbb{E}_{\mathbf{w}_k}\left[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + \left(A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k\right)^T P_{k+1}\left(A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k\right)\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\mathbb{E}_{\mathbf{w}_k}\left[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + \left(A_k\mathbf{x}_k + B_k\mathbf{u}_k\right)^T P_{k+1}\left(A_k\mathbf{x}_k + B_k\mathbf{u}_k\right)\right. \\
&\qquad \left. 2\left(A_k\mathbf{x}_k + B_k\mathbf{u}_k\right)^T P_{k+1}\mathbf{w}_k + \mathbf{w}_k^T P_{k+1}\mathbf{w}_k\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + \left(A_k\mathbf{x}_k + B_k\mathbf{u}_k\right)^T P_{k+1}\left(A_k\mathbf{x}_k + B_k\mathbf{u}_k\right) + \mathrm{tr}\left(P_{k+1}\Sigma_{\mathbf{w}_k}\right)\right)
\end{aligned}
$$

- The optimal cost to go is increased by some constant related to the magnitude of the noise (on which we have no control on)
- The optimal policy is the same as in the deterministic case

Infinite Horizon MDPs

State: $x \in \mathcal{X}$

Action: $u \in \mathcal{U}$

Transition function / Dynamics: $T\left(x_t \mid x_{t-1}, u_{t-1}\right) = p\left(x_t \mid x_{t-1}, u_{t-1}\right)$

Reward function: $r_t = R(x_t, u_t) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$

Discount factor: $\gamma \in (0,1)$

Stationary policy: $u_t = \pi(x_t)$

Typically represented as a tuple

$$\mathcal{M} = (\mathcal{X}, \mathcal{U}, T, R, \gamma)$$

Goal: choose a policy that maximizes cumulative (discounted) reward

$$\pi^* = \arg\max_{\pi} \mathbb{E}_p \left[\sum_{t \geq 0} \gamma^t R\left(x_t, \pi\left(x_t\right)\right) \right]$$

Value functions

State-value function: "*the expected total reward* if we start in that state *and act accordingly to a particular policy*"

$$V_\pi(x) = \mathbb{E}_p\left[\sum_{t\geq 0} \gamma^t R\left(x_t, \pi\left(x_t\right)\right)\right]$$

Action-state value function: "*the expected total reward* if we start in that state, take an action*, and act accordingly to a particular policy*"

$$Q_\pi(x, u) = \mathbb{E}_p\left[\sum_{t\geq 0} \gamma^t R\left(x_t, u_t\right)\right]$$

Optimal state-value function $\quad V^*(x) = \max_\pi V_\pi(x)$

Optimal action-state value function $\quad Q^*(x, u) = \max_\pi Q_\pi(x, u)$

Bellman Equations

Value functions can be decomposed into immediate reward plus discounted value of successor state

$$\mathrm{V}_{\pi}\left(x_t\right) = \mathbb{E}_{\pi}\left[R\left(x_t, \pi\left(x_t\right)\right) + \gamma \mathrm{V}_{\pi}\left(x_{t+1}\right)\right]$$

$$= R\left(x_t, \pi\left(x_t\right)\right) + \gamma \sum_{x_{t+1} \in X} T\left(x_{t+1} \mid x_t, \pi\left(x_t\right)\right) \mathrm{V}_{\pi}\left(x_{t+1}\right)$$

Bellman Expectation Equation

Similarly, also optimal value function can be decomposed as:

Bellman Optimality Equation

$$V^*\left(x_t\right) = \max_u \left(R\left(x_t, u_t\right) + \gamma \sum_{x_{t+1} \in X} T\left(x_{t+1} \mid x_t, u_t\right) V^*\left(x_{t+1}\right) \right)$$

Three paradigms that rely on DP

For *prediction*:

- Policy Evaluation: "given a policy π, find the value function $V_\pi(x)$, i.e., how good is that policy?"

For *control*:

- Policy Iteration: leverages policy evaluation as an inner loop to find the optimal policy

- Value Iteration: applies Bellman's optimality equation to compute the optimal value function

Policy Evaluation

Problem: evaluate a given policy π

Solution: iterative application of Bellman expectation backup ($V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_\pi$)

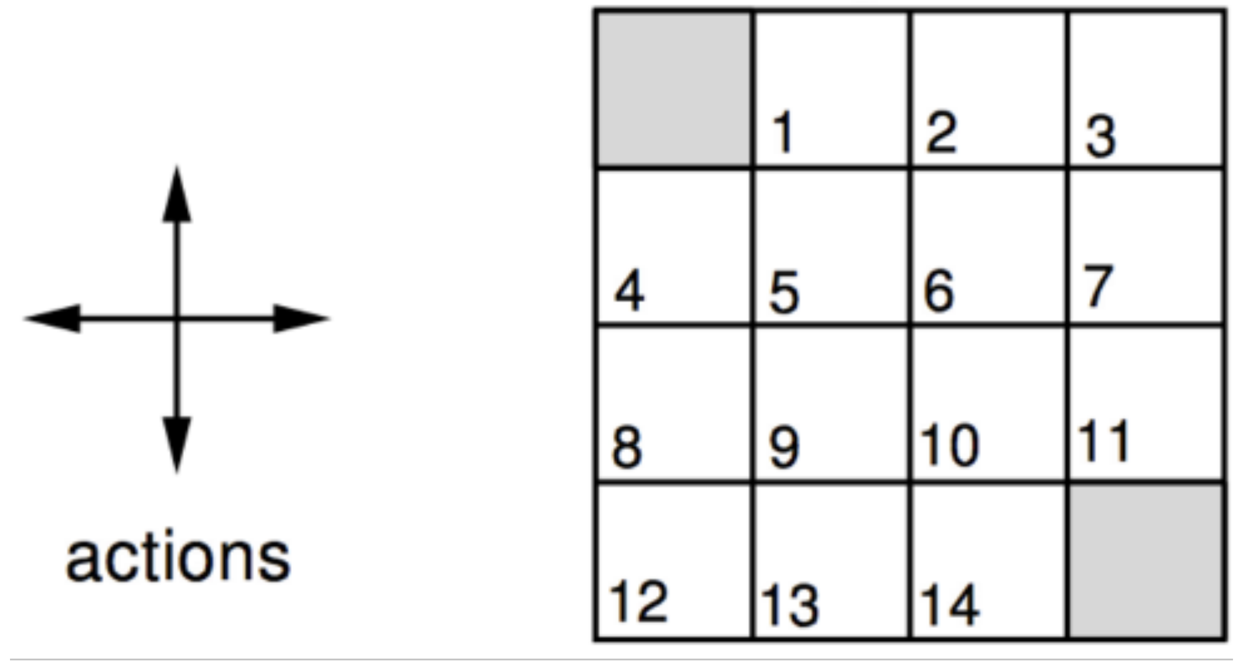
- At each iteration k+1
- For all states x∈X
- Update $V_{k+1}(x)$ from $V_k(x)$ through

<u>**Bellman Expectation Equation**</u>

$$\mathrm{V}_{k+1}\left(x_t\right) = R\left(x_t, \pi\left(x_t\right)\right) + \gamma \sum_{x_{t+1} \in X} T\left(x_{t+1} \mid x_t, \pi\left(x_t\right)\right) \mathrm{V}_k\left(x_{t+1}\right)$$

- This sequence is proven to converge to V_π

Example: Grid World From Sutton and Barto, Reinforcement Learning: An Introduction (Chapter 4)



- Nonterminal states 1, …, 14. Terminal states as shaded squared
- Reward is -1 until the terminal state is reached
- Controls leading out of the grid leave state unchanged
- Undiscounted MDP ($\gamma = 1$)
- We want to evaluate a uniform random policy

$V_k(x)$ for the random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

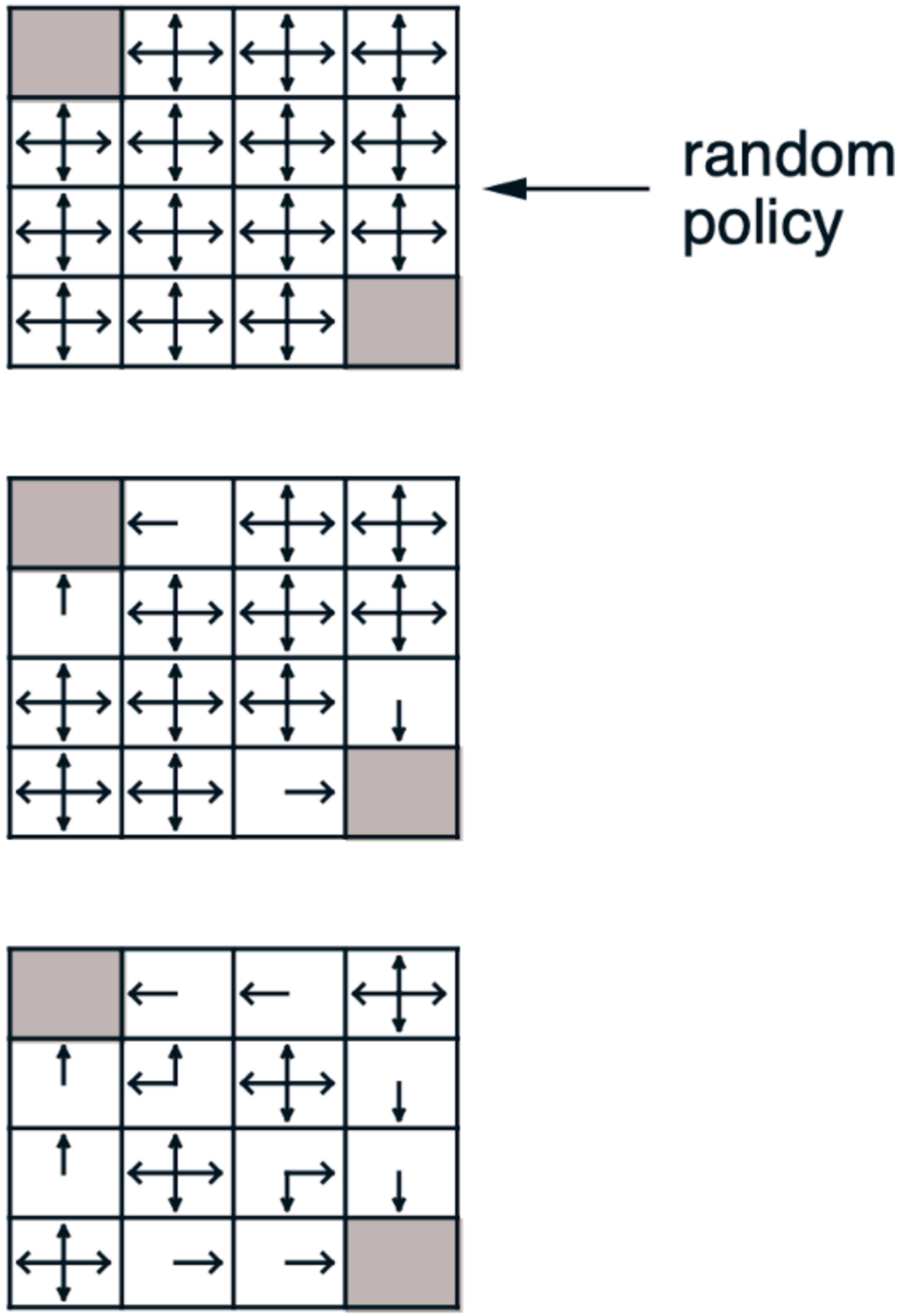
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

Greedy policy w.r.t. $V_k(x)$



$V_k(x)$ for the random policy

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

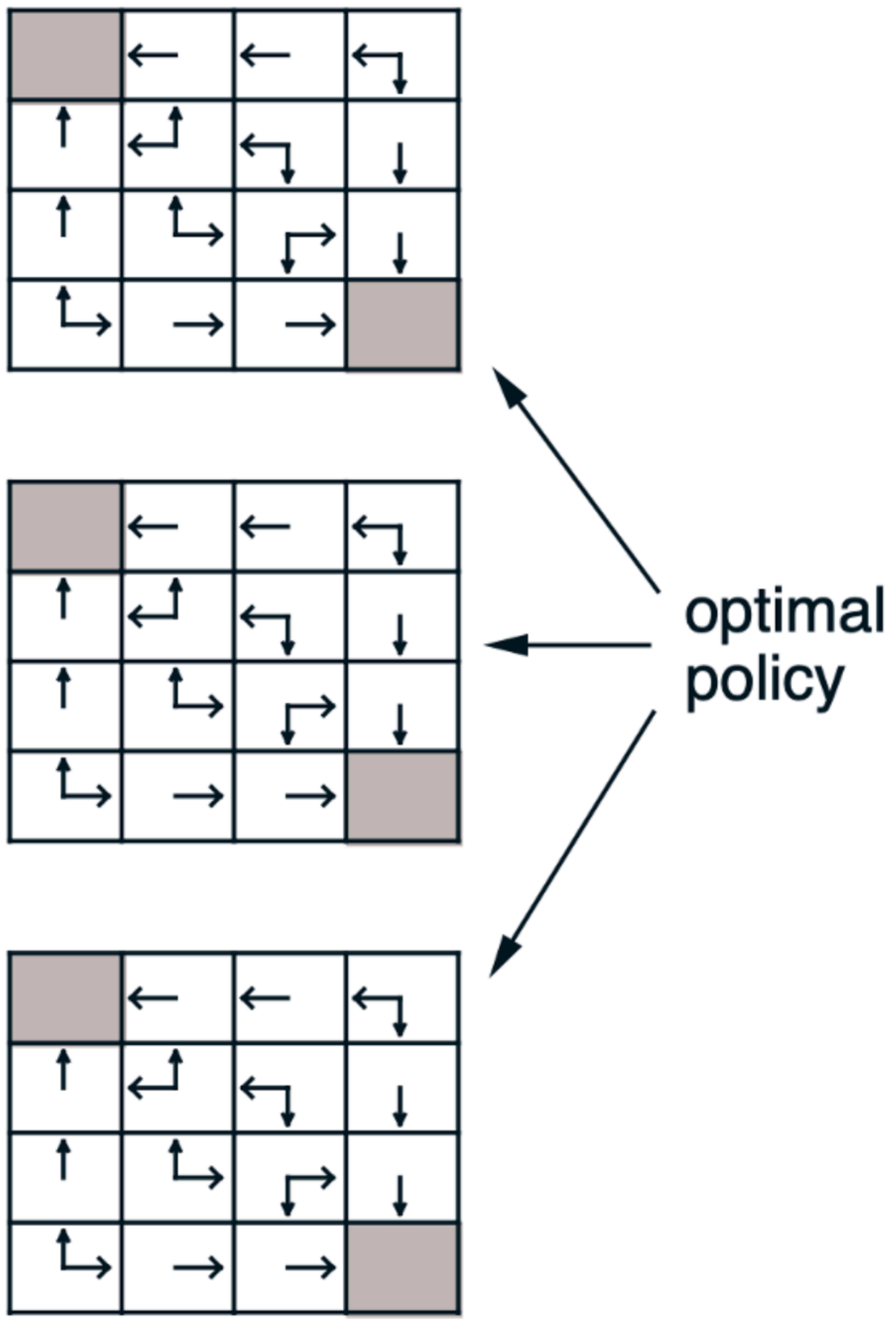
$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Greedy policy w.r.t. $V_k(x)$



Some technical questions

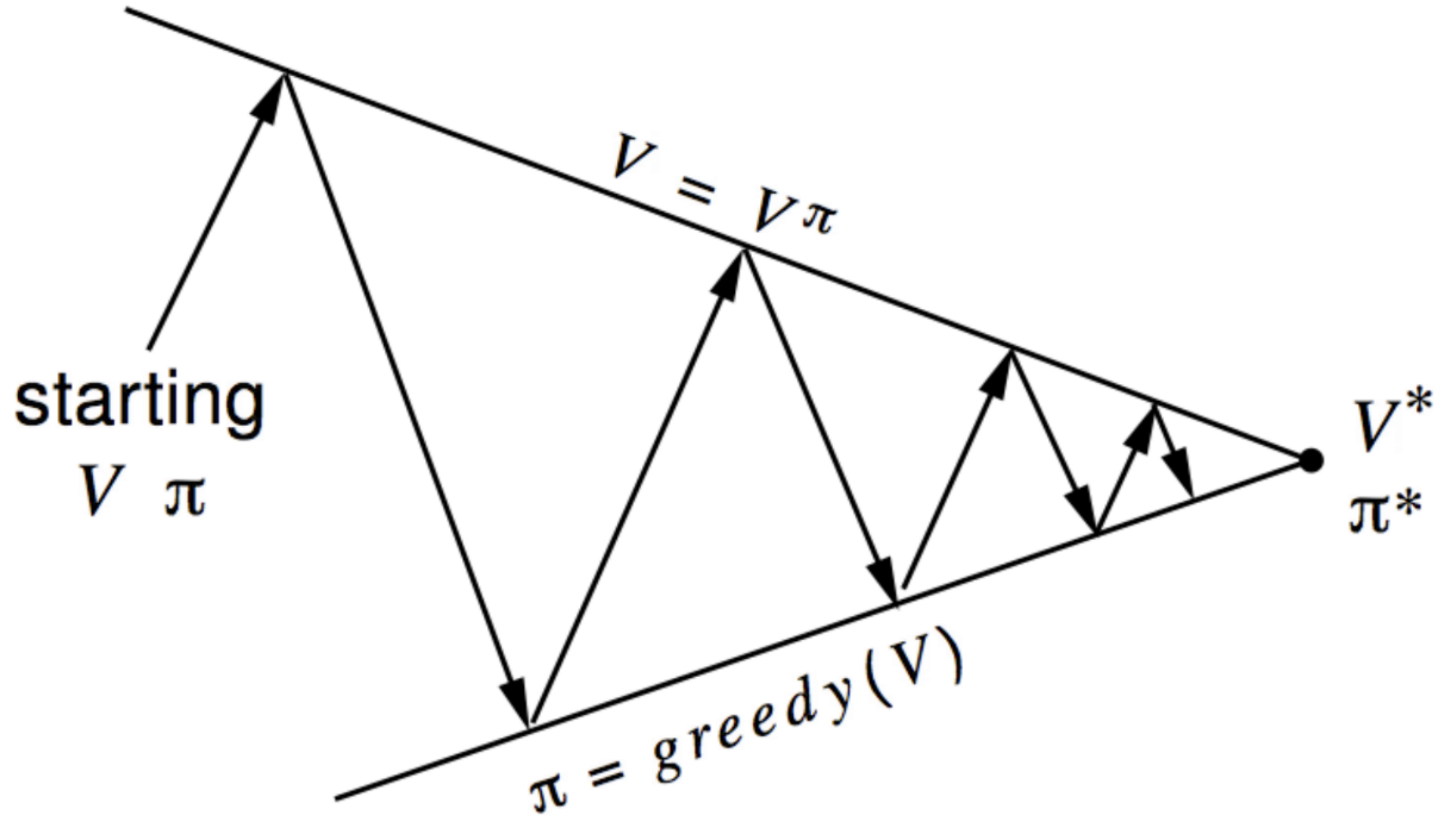
- How do we know that iterative policy evaluation converges to V^{π}?
- Is the solution unique?
- How fast does this algorithm converge?

These questions are resolved by *the contraction mapping theorem*

Sketch of proof:

- Def: ∞-norm $\|\mathrm{u}-\mathrm{v}\|_{\infty}=\max_{x\in\mathcal{X}}|\mathrm{u}(x)-\mathrm{v}(x)|$, i.e. the largest difference between state values
- Def: an update operation is a γ-contraction if $\|U_{i+1}-V_{i+1}|\|\leq\|U_i-V_i\|, \quad \forall U_i, V_i$
- Theorem: a γ-contraction converges to a unique fixed point, no matter the initialization, at a linear convergence rate of γ
- Fact: the policy evaluation operator is a γ-contraction in ∞-norm
- Corollary: policy evaluation converges to a unique fixed point

Policy Iteration



Given policy π

Evaluate the policy π

$$\mathrm{V}_{k+1}\left(x_t\right) = R\left(x_t, \pi\left(x_t\right)\right) + \gamma \sum_{x_{t+1} \in X} T\left(x_{t+1} \mid x_t, \pi\left(x_t\right)\right) \mathrm{V}_k\left(x_{t+1}\right)$$

Improve the policy π by acting greedily w.r.t. V_π

$$\pi_{k+1}(x) = \arg\max_u \left(R(x, u) + \gamma \sum_{x_{t+1} \in \mathcal{X}} T\left(x_{t+1} \mid x_t, u_t\right) V_{k+1}\left(x_{t+1}\right) \right)$$

- In general, policy iteration requires more iterations of evaluation / improvement (in our small Grid World, one was sufficient)
- This process always converges to the optimal policy

Policy Improvement

- Given a deterministic policy $\pi(x)$
- We can improve the policy by acting greedily w.r.t. the current value function

$$\pi'(x) = \underset{u \in \mathcal{U}}{\operatorname{argmax}}\, q_\pi(x, u)$$

- Consider the one step decision, where we use π' for one step and then act accordingly to the old policy π

$$q_\pi\left(s, \pi'(s)\right) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

- If we repeat the same reasoning for all following steps, we can see how this improves the value function $v_{\pi'}(x) \geq v_\pi(x)$

Value Iteration

Problem: find the optimal policy π^*

Solution: iterative application of Bellman optimality backup ($V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_\pi^*$)

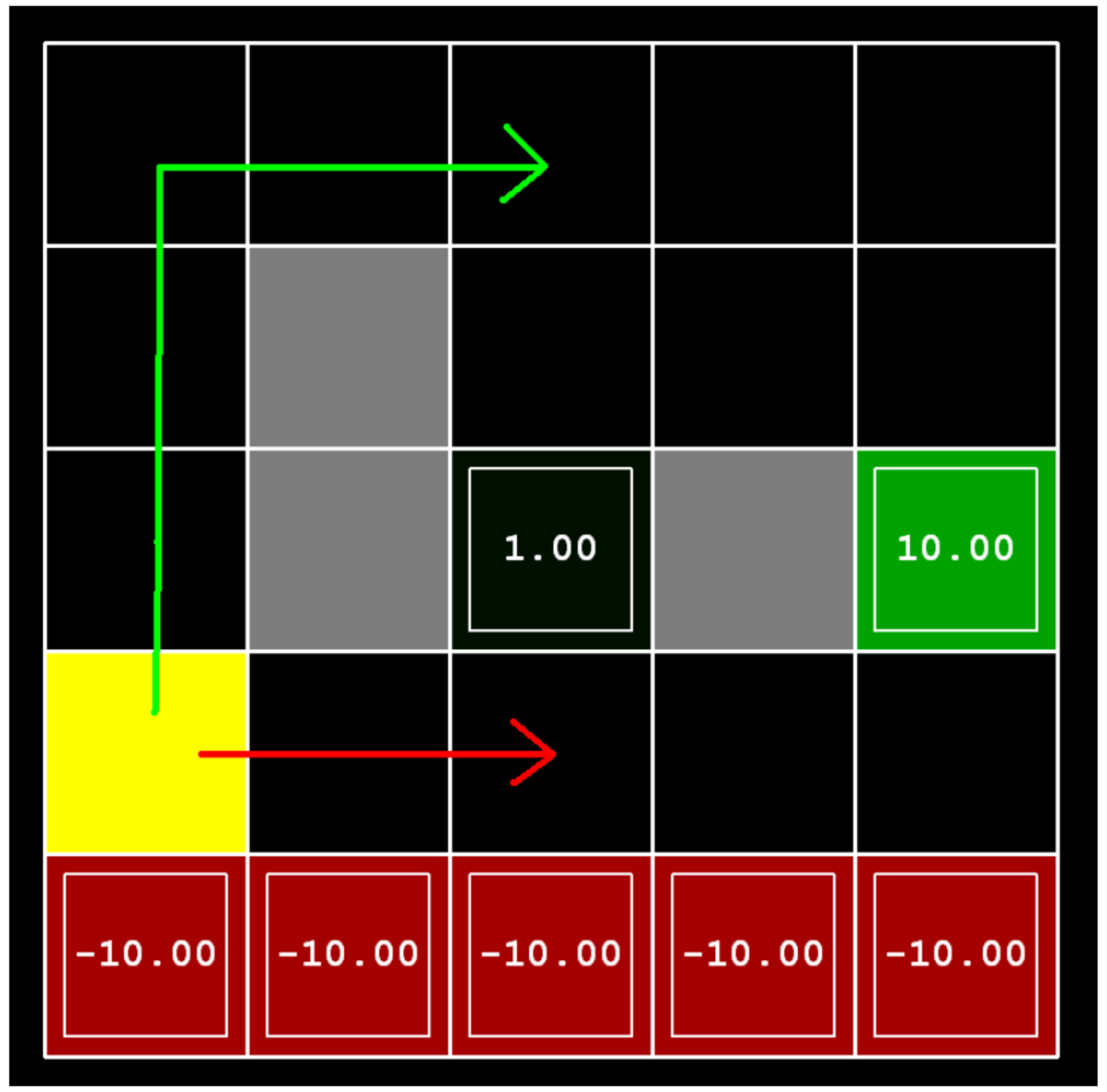
- At each iteration k+1
- For all states x∈X
- Update $V_{k+1}(x)$ from $V_k(x)$ through

Bellman Optimality Equation

$$V_{k+1}^*(x_t) = \max_u \left(R(x_t, u_t) + \gamma \sum_{x_{t+1} \in X} T(x_{t+1} \mid x_t, u_t) V_k^*(x_{t+1}) \right)$$

- This sequence is proven to converge to V^*

Exercise from Pieter Abbeel, CS287



(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

Recap

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

All of these formulations require a **model of the MDP!**

Outline

Stochastic Optimal Control: Markov Decision Process (MDP)

The dynamic programming algorithm (stochastic case)

Stochastic LQR

Infinite-Horizon MDPs:
- Exact Methods:
  - (Policy Evaluation)
  - Value Iteration
  - Policy Iteration

Next time

- Nonlinear LQR for tracking
- iLQR
- DDP