Chapter 4: Dynamic Programming

- DP is a collection of algorithms that can be used to compute optimal policies given a perfect model $p(s',r|s,a)$ of the environment.
- RL does not assume a perfect model.
- Although DP algorithms will not be directly employed in RL, they will lead to some important insights for designing RL algorithms.

Recall Bellman equations for value function and for action-value function:

- $v_\pi(s) = \sum_a \pi(a|s) [\, r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s') \,], \quad s \in \mathcal{S}$
- $q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')\, q_\pi(s',a') \,], \quad s \in \mathcal{S}$
- These are the "fixed-policy" Bellman equations. There are also equations for optimal policies.
- Also recall: $v^*(s) := v_{\pi^*}(s)$ where $\pi^*(s) = \text{argmax}_\pi\, v_\pi(s)$

Some important facts

- $v^*(s) = \max_a \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a) v^*(s') \}$ "Bellman optimality equation"
- $q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} q^*(s',a')$ action-value version
- $v^*(s) = \max_a q^*(s,a)$
- Optimal deterministic policy π^* can be obtained from:

$$\pi^*(s) = \text{argmax}_a \, r(s,a) + \gamma \sum_{s'} p(s'|s,a) v^*(s')$$

or

$$\pi^*(s) = \text{argmax}_a \, q^*(s,a)$$

So if we can determine $v^*(s)$ or $q^*(s,a)$, then we can easily obtain an optimal policy π^*.

Another fact

- Suppose $v_\pi(s)$ satisfies the Bellman optimality equation:

$$v_\pi(s) = \max_a \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s') \}$$

- Then π is an optimal policy.

- **So π is optimal if and only if it satisfies the Bellman optimality equation**

Policy Evaluation

- Let's first consider how we can determine $v_\pi(s)$ for a fixed policy π.

- Recall Bellman equation

$$v_\pi(s) = \sum_a \pi(a|s) \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s') \right], \quad s \in \mathcal{S}$$

- System of $|\mathcal{S}|$ linear equations with $|\mathcal{S}|$ unknowns. Can be solved with standard algorithms from linear algebra

- Fixed-policy iterative algorithm: Initialize $v_0(s)$, $s \in \mathcal{S}$, arbitrarily. Then repeat:

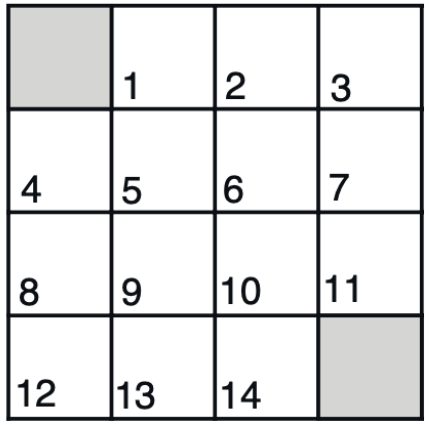
$$v_{k+1}(s) \leftarrow \sum_a \pi(a|s) \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_k(s') \right] \qquad s \in \mathcal{S}$$

- Can be shown that $v_k(s)$ converges to $v_\pi(s)$ (contraction mapping theorem)

Goal: Reach terminal state as quickly as possible.

Example 4.1 Consider the 4×4 gridworld shown below.

actions



$R_t = -1$
on all transitions

The nonterminal states are $\mathcal{S} = \{1, 2, \ldots, 14\}$. There are four actions possible in each state, $\mathcal{A} = \{\texttt{up}, \texttt{down}, \texttt{right}, \texttt{left}\}$, which deterministically cause the corresponding state transitions, except that actions that would take the agent off the grid in fact leave the state unchanged. Thus, for instance, $p(6, -1 \mid 5, \texttt{right}) = 1$, $p(7, -1 \mid 7, \texttt{right}) = 1$, and $p(10, r \mid 5, \texttt{right}) = 0$ for all $r \in \mathcal{R}$. This is an undiscounted, episodic task. The reward is -1 on all transitions until the terminal state is reached. The terminal state is shaded in the figure (although it is shown in two places, it is formally one state). The expected reward function is thus $r(s, a, s') = -1$ for all states s, s' and actions a. Suppose the agent follows the equiprobable random policy (all actions equally likely).

When there are terminal states, need to initialize $v_0(s) = 0$ for all terminal states.

In this example, authors set $v_0(s) = 0$.

If s is not a terminal state:

$v_{k+1}(s) \leftarrow \sum_a (1/4)[\ -1 + \gamma \sum_{s'} p(s'|s,a)\ v_k(s')\ ] = -1$

v_k 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

$k = 3$

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

$= v_\pi(s)$

Policy Improvement

- Now that we can evaluate a policy π, how do we improve it?

Policy improvement theorem: Let π and π' be any pair of deterministic policies. Suppose

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \text{ for all } s \in \mathcal{S}. \qquad (4.7)$$

then

$$v_{\pi'}(s) \geq v_\pi(s) \text{ for all } s \in \mathcal{S} \qquad (4.8)$$

Moreover, if there is strict inequality in (4.7) for at least one state, then there will be strict inequality in (4.8) for at least one state.

Restatement of Policy Improvement Theorem

Policy improvement theorem: Let π be any policy. Let $\pi'(s) = \text{argmax}_a\, q_\pi(s,a)$ for all $s \in \mathcal{S}$. Then $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$.

Furthermore if $\pi'(s) = \pi(s)$ for all $s \in \mathcal{S}$, then π ($= \pi'$) is optimal.

Policy iteration action algorithm

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

Begin with some policy π

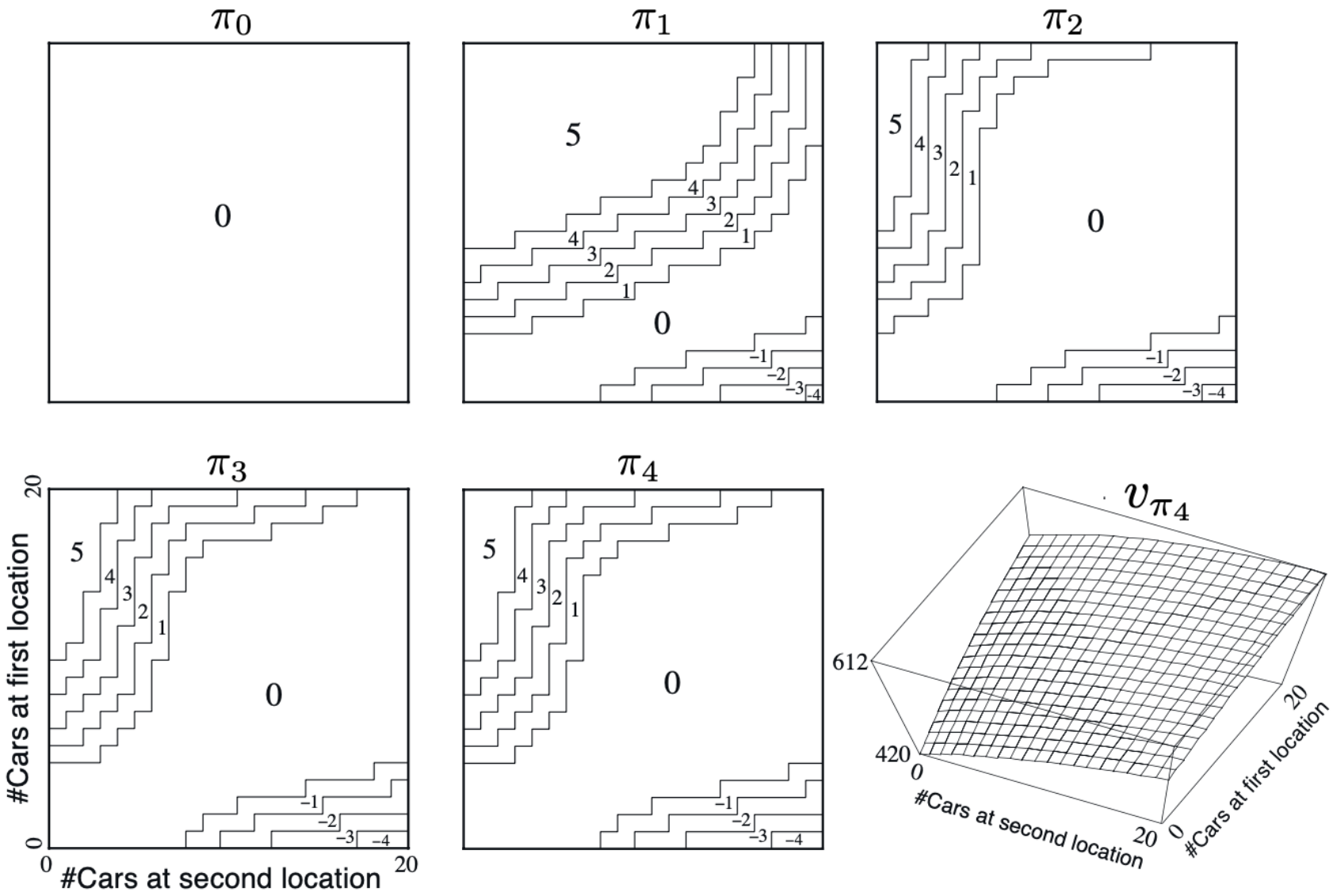
Repeat:

- Evaluate $v_\pi(s)$, $s \in \mathcal{S}$, using the fixed-policy iterative algorithm.
- Improve the policy using:

$$\pi(s) \leftarrow \text{argmax}_a \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s') \} \qquad s \in \mathcal{S}$$

Keep repeating until π(s) does not change for all $s \in \mathcal{S}$.

Optimal policy for Jack's rental car



Value Iteration

Repeat:

$$v_{k+1}(s) = \max_a \{r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_k(s')\}, \qquad s \in \mathcal{S}$$

Theorem: v_k converges to v^* as $k \to \infty$.

Can be thought of as an improvement sweep followed by an evaluation sweep:

$$\pi(s) = \text{argmax}_a \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_k(s') \} \qquad s \in \mathcal{S} \qquad \text{improve sweep}$$

$$v_{k+1}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s,\pi(s)) v_k(s') \qquad s \in \mathcal{S} \qquad \text{evaluate sweep}$$

$v_{k+1}(s)$ is a very rough approximation of $v_\pi(s)$. Could do more evaluation sweeps with π fixed to get a better approximation of $v_\pi(s)$.

Generalized Policy Iteration

- Alternate between approximate policy evaluation and approximate policy improvement:
  - Do not necessarily evaluate policy fully during evaluation step. Instead drive it a little towards the value function of the policy
  - In which case have approximate value function
  - Then improve policy wrt approximate value function
- The value iteration algorithm (previous slide) is GPI. Also leads to optimal policy.
- Most DRL algorithms are GPI but provide only approximate optimal policies.