Reinforcement Learning

Lecture 4

Planning: Control

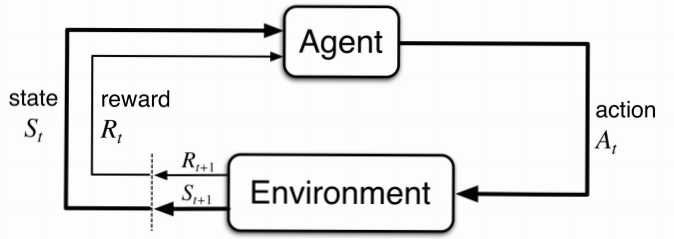
Greedy Policy, Policy Improvement Theorem

Policy Iteration

Robert Peharz

Institute of Theoretical Computer Science
Graz University of Technology
Winter Term 2023/24

Recap



Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, p)$

state space $\mathcal{S}$, action space $\mathcal{A}$, dynamics $p(s', r \mid s, a)$

Policy $\pi(a \mid s)$

Discounted Return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad , \quad \gamma \in [0, 1]$$

Value Function

$$v_\pi(s) := \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

Recap

Bellman Expectation Equation

Image: wikipedia.org

$$v_\pi(s) = \mathbb{E}_{A_t, R_{t+1}, S_{t+1}} \left[R_{t+1} + \gamma \, v_\pi(S_{t+1}) \mid S_t = s \right]$$

$$v_\pi(s) = r(s) + \gamma \sum_{s'} p(s'|s) \, v_\pi(s') \qquad \forall s \in \mathcal{S}$$

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

Matrix-Vector form:

$$\underline{v} = \begin{pmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ \vdots \\ v_\pi(s_N) \end{pmatrix} \quad \underline{r} = \begin{pmatrix} r(s_1) \\ r(s_2) \\ \vdots \\ r(s_N) \end{pmatrix} \quad P = \begin{pmatrix} p(s_1|s_1) & \cdots & p(s_N|s_1) \\ \vdots & \ddots & \vdots \\ p(s_1|s_N) & \cdots & p(s_N|s_N) \end{pmatrix}$$

$$\underline{v} = \underline{r} + \gamma P \underline{v} \quad \rightarrow \text{Analytic solution:} \quad \underline{v}_\pi = (I - \gamma P)^{-1} \underline{r}$$

Recap

Iterative Policy Evaluation

Bellman equation
(interpreted as update rule)

Iterative Policy Evaluation

- initialize $V(s)$ arbitrarily (e.g. all 0)
- repeat
  - for $s \in \mathcal{S}$ do $V_{new}(s) \leftarrow r(s) + \gamma \sum_{s'} p(s'|s)\, V(s')$
  - if $\forall s: V_{new}(s) \approx V(s) \rightarrow$ break
  - $V \leftarrow V_{new}$

Random walk policy $\hat{\pi}(a|s) = 0.25$ $(\leftarrow, \rightarrow, \uparrow, \downarrow)$

k=0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

k=1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

k=2

0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0

...

k=10

0	-6.1	-8.3	-8.9
-6.1	-7.7	-8.4	-8.3
-8.3	-8.4	-7.7	-6.1
-8.9	-8.3	-6.1	0

...

k=199

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

k=200

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

$\approx V_{\hat{\pi}}$

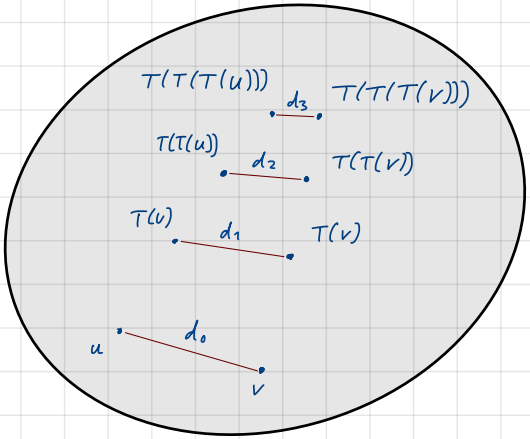
Recap

Contraction (Contraction mapping)

Let V be any vector space. An L-Lipschitz function $T: V \to V$ with $L < 1$ is called a **contraction** (L-contraction).

Fixed point

Let V be some vector space and let $T: V \to V$ be a function. A vector $v \in V$ is called a **fixed point** of T if $v = T(v)$, i.e. if T maps v onto itself.



Banach's Fixed Point Theorem

Let V be some vector space and let $T: V \to V$ be a contraction. Then T has a unique fixed point $v^* \in V$. Moreover, for any $v_0 \in V$, the sequence defined via

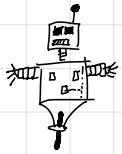
$$v_n = T(v_{n-1})$$

converges to v^*, i.e. $v_n \to v^*$ as $n \to \infty$.

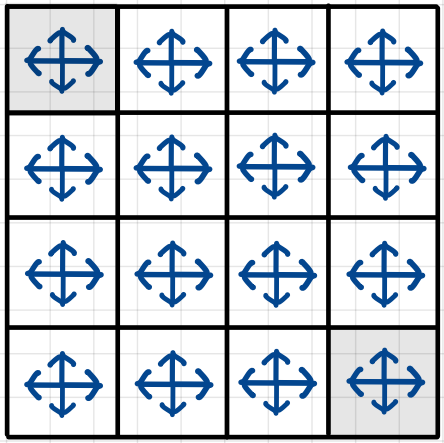
→ theoretical underpinning for many RL algorithms !
(needed for your homework)

How to Improve Policies?

Policy:



$\pi(a|s) \equiv 0.25$



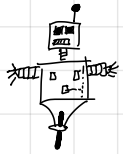
Value function V_π

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

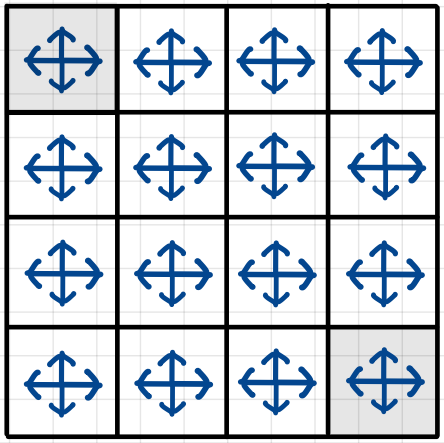
Let's help the robot!

- the value function tells us "how good" a state s is under current policy π
- assume any state s
- assume you could "hack" the agent's policy for one time step, i.e. you can select the agent's next action
- after that, the agent resumes its policy
- which action do you pick?

How to Improve Policies?

Policy: 

$\pi(a|s) \equiv 0.25$

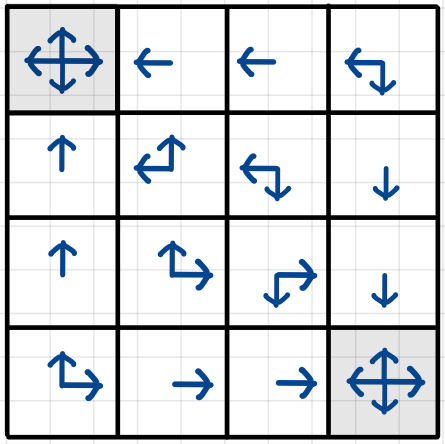


Value function V_π

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

better: larger value

- we should move the agent to better states

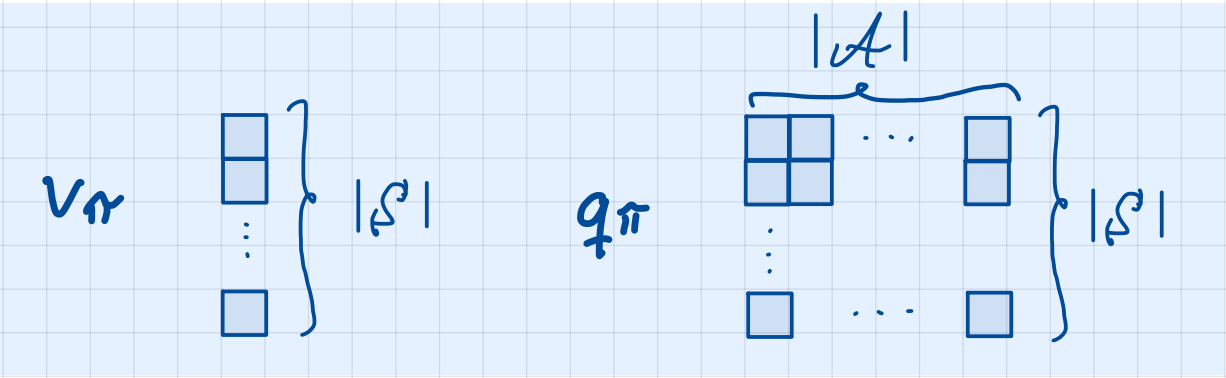


(multiple arrows denote equally good actions)

- we actually just came up with a new policy! (**greedy policy**)

- is it always better than the old one?

The q Function



- to formalize greedy policies we introduce the **q-function**
- we have previously defined the value function

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

- the **q-function** is the value function where we first pick an action of our choice:

$$q_\pi(s,a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

- also called **state-action value function**.

Relation Between v and q

We can express v_π via q_π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_{A_t}\Big[\overbrace{\mathbb{E}_\pi[G_t \mid s, A_t]}^{q(s, A_t)} \mid S_t = s\Big] = \sum_a \pi(a|s)\, q_\pi(s,a) \tag{1}$$

We can express q_π via v_π:

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi\Big[\overbrace{R_{t+1} + \gamma G_{t+1}}^{= G_t} \mid S_t = s, A_t = a\Big]$$

$$= \underbrace{r(s,a)}_{\text{expected reward}} + \gamma\, \mathbb{E}_{s_{t+1}}\Big[\overbrace{\mathbb{E}_\pi[G_{t+1} \mid \cancel{S_t = s}, \cancel{A_t = a}, S_{t+1}]}^{= v_\pi(s_{t+1})} \mid S_t = s, A_t = a\Big] \quad \text{Markov}$$

$$= r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s') \tag{2}$$

Bellman Expectation Equation in q

Plugging (1) into (2) yields the **Bellman equation** in q:

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

How to compute q_π:

given by MDP

Option 1: Compute v_π and let $q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \, v_\pi(s')$

Option 2: Directly solve Bellman equation in q

- closed form (matrix inversion)
- iterating Bellman equation with arbitrary initial q_0

Greedy Policy

- let π be any policy and q_π its q-function
- define a new <u>deterministic</u> policy π':

$$\pi'(a|s) := \begin{cases} 1 & \text{if } a = \arg\max_{a'} q_\pi(s,a') \\ 0 & \text{otherwise} \end{cases}$$

(pick arbitrary $a' \in \arg\max$ if maximizer is not unique)

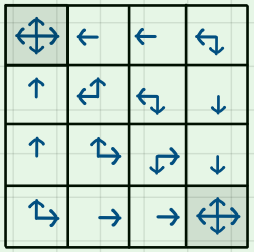
$\pi'(a|s)$ is called the **greedy policy** w.r.t. q_π (v_π)

Example: Grid World

Value function

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

Greedy Policy



(multiple arrows : non-unique argmax)

is it better than the old one?

Policy Improvement Theorem

Let π be any policy and π' the (a) greedy policy w.r.t. v_π (q_π). Then

$$v_\pi(s) \le v_{\pi'}(s) \qquad \forall s \in \mathcal{S}$$

Recall: $v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)$ $\qquad$ $q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s')$

$$v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)$$

$$\le \max_a q_\pi(s,a) \qquad \text{// use } \pi' \text{ once, then } \pi$$

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

$$\le \max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} q_\pi(s',a') \qquad \text{// use } \pi' \text{ twice, then } \pi$$

$$\le \max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left(\max_{a'} r(s',a') + \gamma \sum_{s''} p(s''|s',a')\, v_\pi(s'') \right)$$

$$\le \cdots \le v_{\pi'}(s) \qquad \text{// by induction, } \pi' \text{ is at least as good as } \pi$$

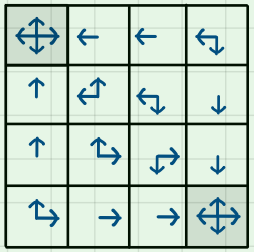
□

Example: Grid World

Value function

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

Greedy Policy



(multiple arrows : non-unique argmax)

is it better than the old one?

Yes!

A greedy policy is always at least as good as the original policy.

- Note that a greedy policy is deterministic
- Thus, deterministic policies are "enough" *
- We can now iterate the "greedyfication" → policy iteration

---

* For planning (known MDP) — for general RL, stochastic policies will be important for exploration.

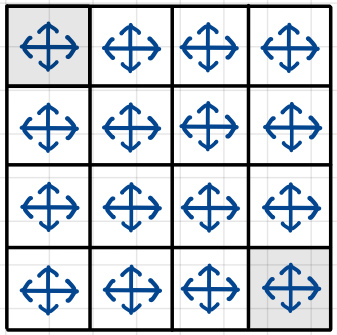
Policy Iteration

- initialize $\hat{\pi}_0$, $K \leftarrow 0$
- repeat
  - (1) compute $q_{\hat{\pi}_K}$ // e.g. with iterative policy evaluation
  - (2) let $\hat{\pi}_{K+1}(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a'} q_{\hat{\pi}_K}(a'|s) \\ 0 & \text{otherwise} \end{cases}$ // greedy policy
  - if $q_{\hat{\pi}_{K+1}} \approx q_{\hat{\pi}_K}$ $\longrightarrow$ break
  - $K \leftarrow K+1$

Example: Grid World

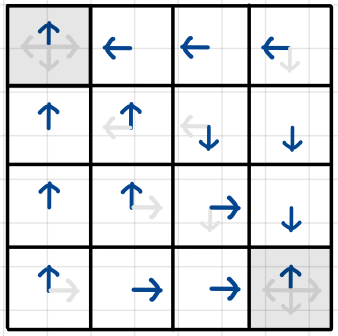
π_k | V_{π_k} $\left(q_{\pi_k} = r(s,a) + \gamma\, \mathbb{E}_{s_{t+1}}[v_{\pi_k}(s_{t+1})]\right)$

$k=0$



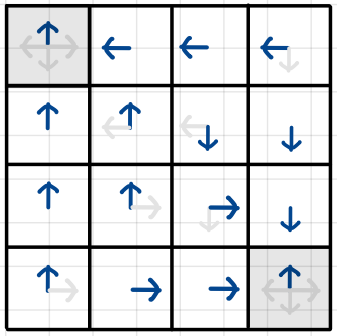
0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0

$k=1$

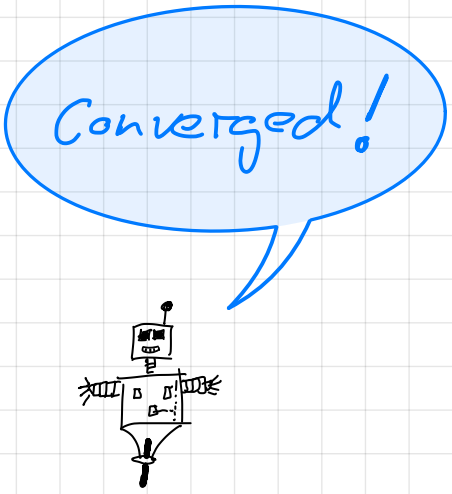


0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

$k=2$



0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0



This policy seems **globally optimal!**

Does this always happen?

Optimal Value Function

- What does it mean for a policy to be optimal?

  Intuitively, a policy π^* is optimal if $V_{\pi^*}(s) \geq V_{\pi}(s)$, $\forall \pi, s \in \mathcal{S}$

- Does an optimal policy always exist?

  (Not evident: They might be incomparable, meaning for any two π_1, π_2, there could be $s, s' \in \mathcal{S}$, such that $V_{\pi_1}(s) > V_{\pi_2}(s)$ and $V_{\pi_1}(s') < V_{\pi_2}(s')$)

- First, let's define **optimal value functions**

$$V^*(s) := \max_{\pi} V_{\pi}(s)$$

$$q^*(s, a) := \max_{\pi} q_{\pi}(s, a)$$

— upper bound for all V_π, q_π, since $\max_\pi$ is taken separately for all s, a.

- If there was a π^* with $V_{\pi^*} = V^*$ then π^* would be optimal.

Bellman Optimality Equation

$$\pi'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a'} q_\pi(s,a') \\ 0 & \text{otherwise} \end{cases}$$

When $v_\pi = v_{\pi'}$, then policy iteration has converged.

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) \mid S_t = s] = v_{\pi'}(s)$$

$$v_\pi(s) = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

$$= \sum_a \pi'(a|s) \Big[\underbrace{r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s')}_{q_\pi(s,a)} \Big]$$

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

Bellman Optimality Equation

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

Holds, when policy iteration has converged

Bellman Optimality Equation (in q)

Policy iteration converged: $q_{\hat{\pi}} = q_{\hat{\pi}'}$

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

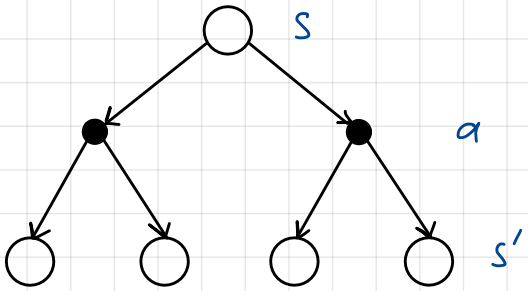
$$= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \hat{\pi}'(a'|s') q_{\hat{\pi}'}(s',a') = q_{\hat{\pi}'}(s,a)$$

Bellman Optimality Equation (in q)

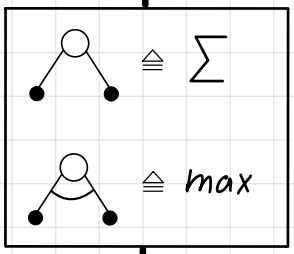
$$q_{\hat{\pi}}(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} q_{\hat{\pi}}(s',a')$$

Backup Diagrams: Graphical Descriptions of RL Algorithms

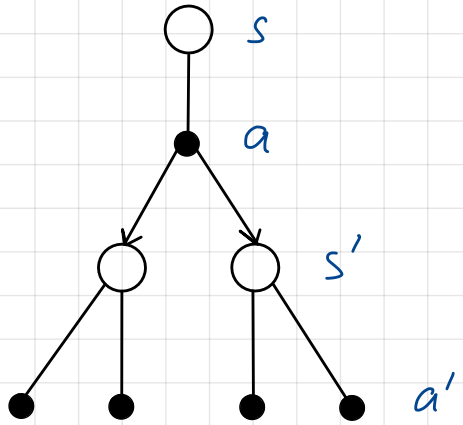
Bellman Expectation Equation in v



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

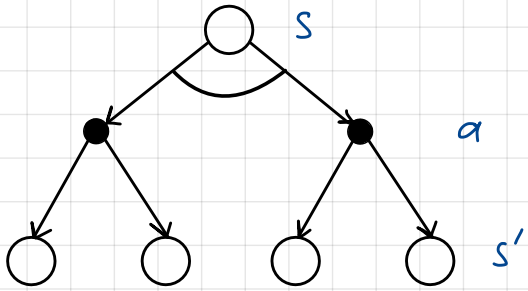


Bellman Expectation Equation in q



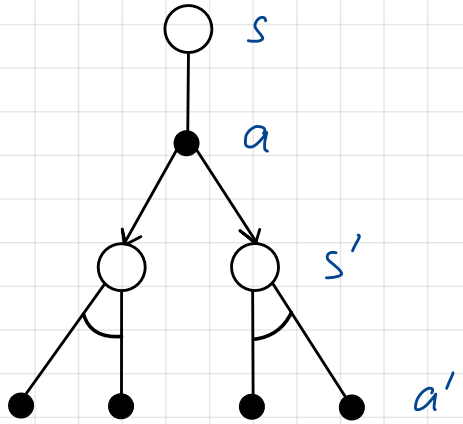
$$q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')\, q(s',a')$$

Bellman Optimality Equation in v



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

Bellman Optimality Equation in q



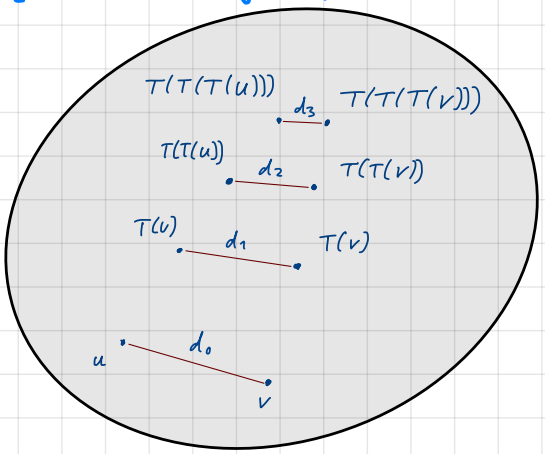
$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} q_\pi(s',a')$$

Theorem: Global Optimal Policy

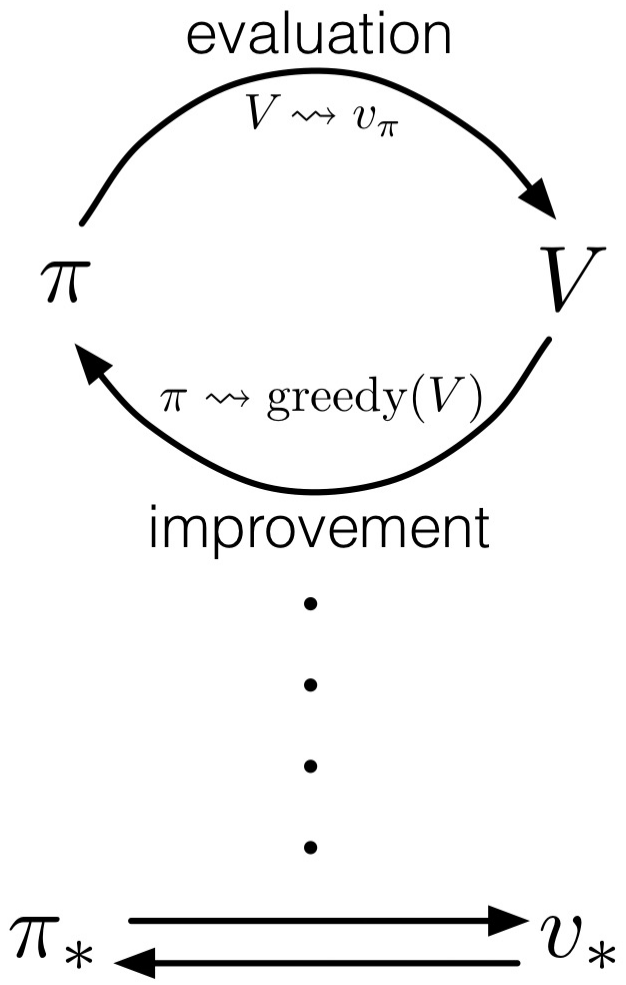
For any initial policy π_0, policy iteration converges to an optimal policy π^* with value function V^*

Proof sketch (see Szepesvári, page 78)

- When Policy Iteration has converged, the Bellman Optimality Equation holds (why will it converge?)
- Show that Bellman Optimality Equation is a contraction
- Hence, it converges to a unique fixed point
- Show that this fixed point is an upper bound of V^*
- But the fixed point is also $V_{\pi'}$ of the learned π' — why?
- Hence V^* must be the value function of π'



Schematic of Policy Iteration



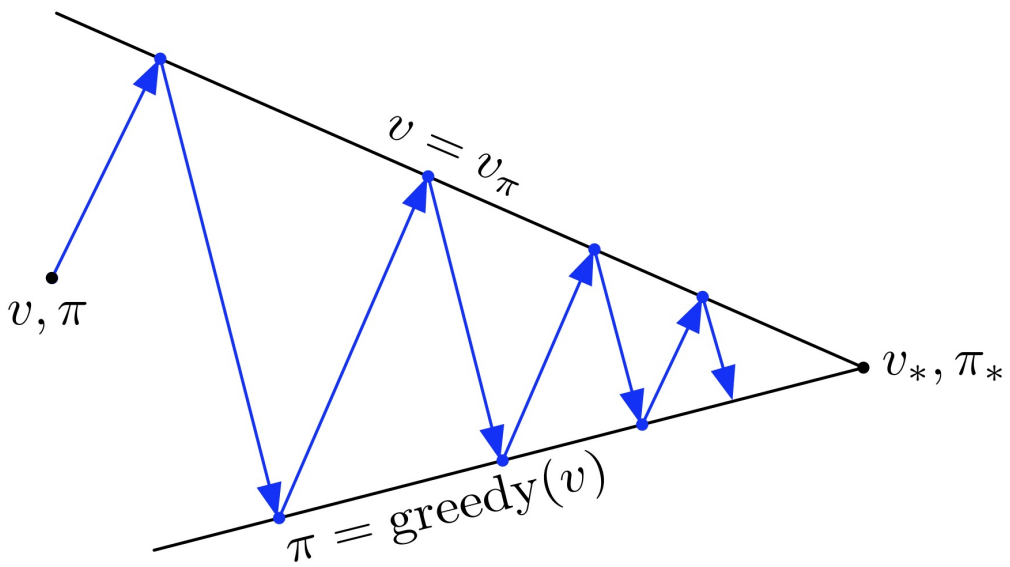


Image: Sutton & Barto

Car Rental

(Sutton & Barto, p. 81)

- a manager is in charge of two car rental stations
- if a customer requests a car at a station:
  - if car available → rented out for 10 €
  - otherwise → 0 €
- returned cars get available the next day
- maximal 20 cars at each station (additional cars returned elsewhere)
- numbers of requested and returned cars are Poisson distributed

$$p(n) = \frac{\lambda^n}{n!} e^{-\lambda}$$

station 1: $\lambda_{request} = 3$ $\lambda_{return} = 3$

station 2: $\lambda_{request} = 4$ $\lambda_{return} = 2$

(i.e., typically, more cars are requested at station 2 than returned)

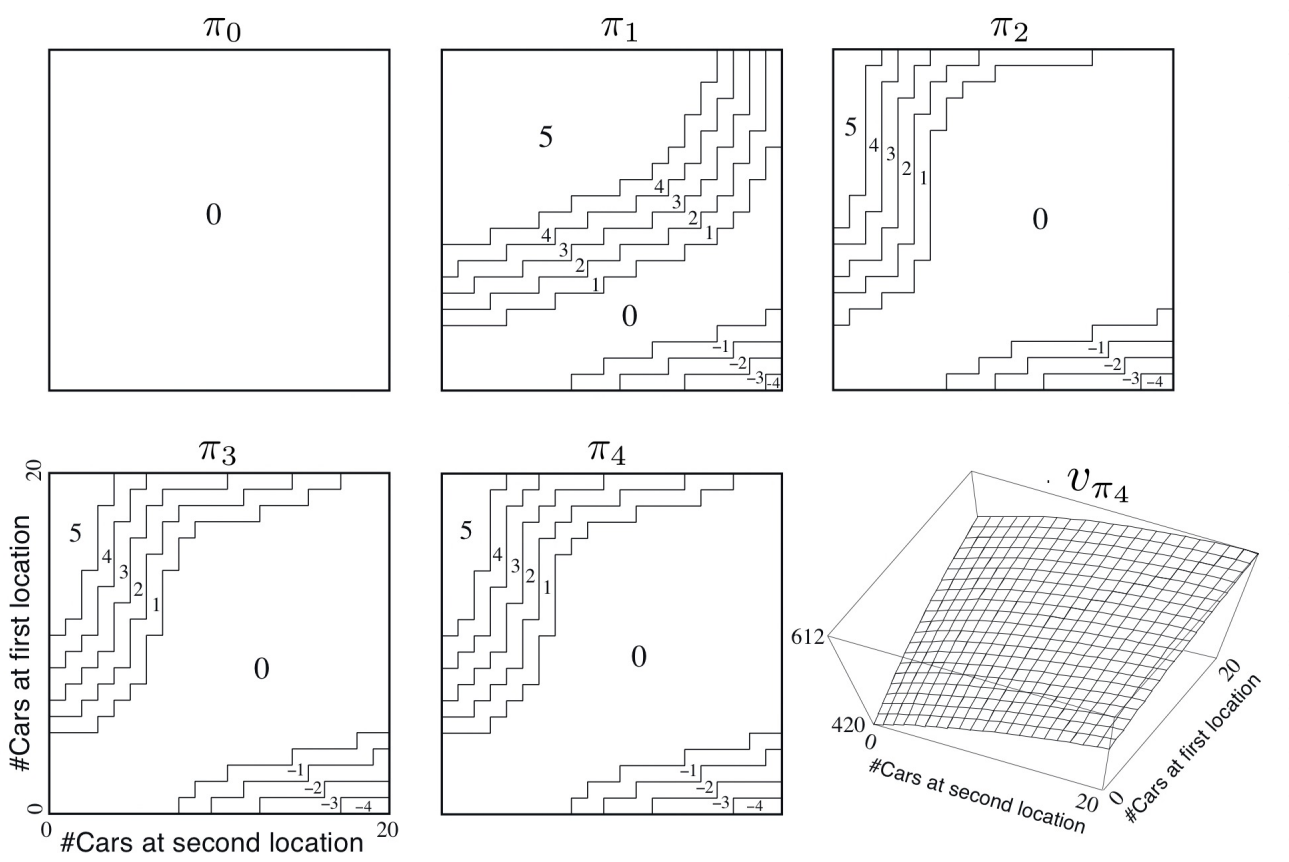
- manager can transfer up to 5 cars between stations each day for the cost of 2 €

Car Rental cont'd

(Sutton & Barto, p. 81)

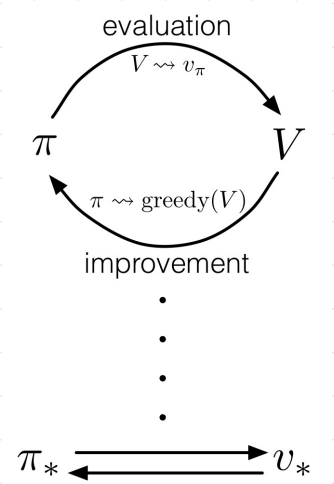
- state space $\mathcal{S} = \{0, 1, 2, \ldots, 20\} \times \{0, 1, 2, \ldots, 20\}$ (cars at station 1 × cars at station 2)
- action space $\mathcal{A} = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$ (number of cars moved from station 1 to station 2)
- $p(s'|s,a)$ and $r(s,a)$ governed by Poisson distributions

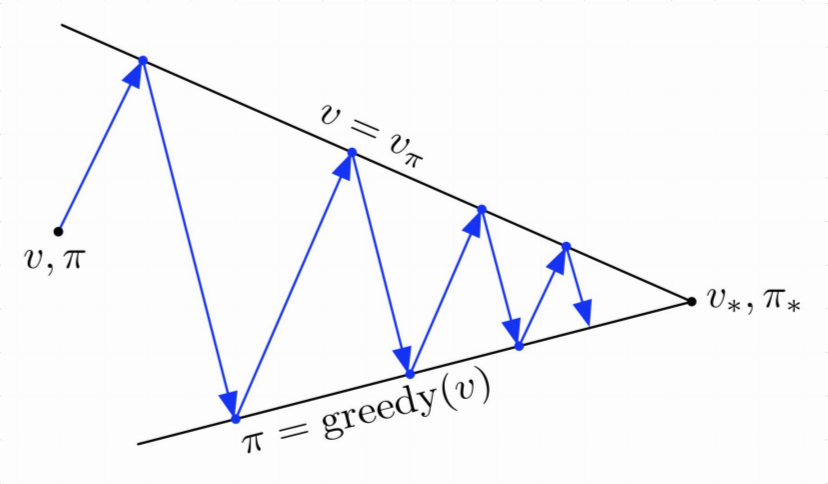
Policy iteration:



Efficiency of Policy Iteration?

Image: Sutton & Barto





- policy evaluation is expensive (in particular closed form)
- Iterative Policy Evaluation only converges for $n \to \infty$

  before that, intermediate results are typically not even a value function of any policy !
- how many iterations of Iterative Policy Evaluation do we really need?

Grid World

$\pi(a|s) = 0.25$

Iterative Policy Evaluation

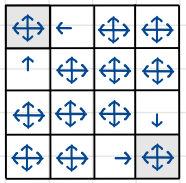
Greedy Policy

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

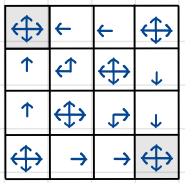
$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0



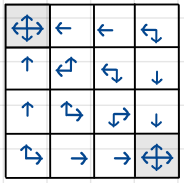
$k = 2$

0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0



$k = 3$

0	-2.4	-2.9	-3
-2.4	-2.8	-3	-2.9
-2.9	-3	-2.8	-2.4
-3	-2.9	-2.4	0

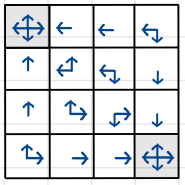


. . .

. . .

$k = 10$

0	-6.1	-8.3	-8.9
-6.1	-7.7	-8.4	-8.3
-8.3	-8.4	-7.7	-6.1
-8.9	-8.3	-6.1	0

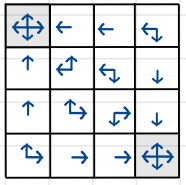


. . .

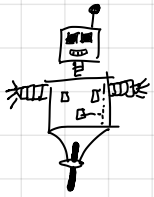
. . .

$k = \infty$

0	-13.8	-19.6	-21.6
-13.8	-17.7	-19.6	-19.6
-19.6	-19.6	-17.7	-13.8
-21.6	-19.6	-13.8	0



Optimal policy!



Policy Iteration with Truncated Policy Evaluation

initialize $\hat{\pi}$, v

repeat until convergence

  repeat for K steps

    $\forall s: \quad v(s) \leftarrow r(s) + \gamma \sum_a \hat{\pi}(a|s) \sum_{s'} p(s'|s,a)\, v(s')$

  $\hat{\pi} \leftarrow \text{greedy}(v)$

Value Iteration

Special case for $k=1$

initialize $\hat{\pi}$, V

repeat until convergence

$$\forall s: \quad V(s) \leftarrow r(s) + \gamma \sum_a \hat{\pi}(a|s) \sum_{s'} p(s'|s,a)\, V(s')$$

$$\hat{\pi}'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a'} \underbrace{r(s,a') + \gamma \sum_{s'} p(s'|s,a)\, V(s')}_{q(s,a)} \\ 0 & \text{o.w.} \end{cases}$$

$$V^{new}(s) = \underbrace{\sum_a \hat{\pi}'(a|s)\, r(s,a)}_{r(s)} + \gamma \sum_a \hat{\pi}'(a|s) \sum_{s'} p(s'|s,a)\, V^{old}(s')$$

$$= \sum_a \hat{\pi}'(a|s) \left(r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, V^{old}(s') \right)$$

$$= \max_a \; r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, V^{old}(s')$$

This is the iterating the Bellman Optimality Equation !

Theorem: Global Optimal Policy showed that this converges to V^* and $\hat{\pi}^*$.

Value Iteration

Policy Iteration with Truncated Policy Evaluation

Policy Iteration

1 — number k of policy evaluation steps — ∞

Converges to v^* and π^* for all k!

Optimal k, with least computation? Not easy to determine.

Asynchronous Updates

...

repeat until convergence

  repeat for K steps

    $\forall s: \quad V(s) \leftarrow r(s) + \gamma \sum_a \hat{\pi}(a|s) \sum_{s'} p(s'|s,a) V(s')$

...

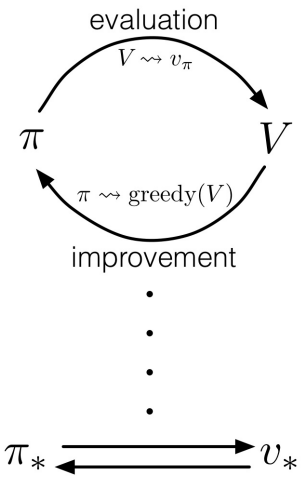
Do we have to always update all state values?

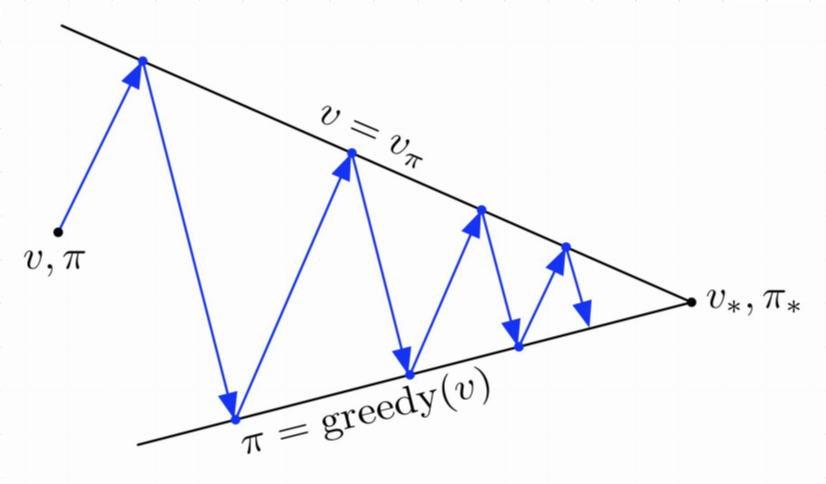
- If number of states is large, one iteration takes very long
- We could use updated values immediatly

No.

- We can update subsets of $\mathcal{S}$ (also single s), in any order
- Convergence to $V^*, \hat{\pi}^*$ if
  - all states are updated eventually
  - infinitely often

Generalised Policy Iteration





- Truncated policy evaluation & asynchronous updates approximate exact policy evaluation
- **Generalized policy iteration**: any scheme which moves
  - v closer to $v_{\hat{\pi}}$
  - $\hat{\pi}$ closer to greedy(v)

Typically converges to $v^*, \hat{\pi}^*$ (shown via contraction arguments)