Reinforcement Learning

Markov Decision Processes: Part 2 + Dynamic Programming: Part 1

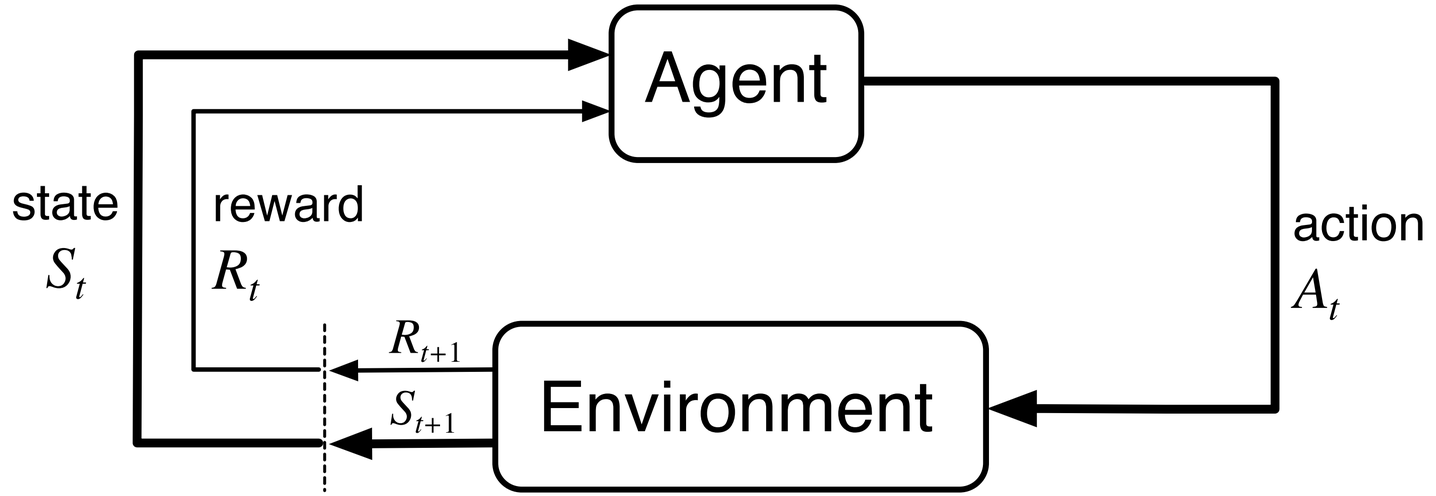
David Abel, Michael Herrmann
Based on slides by Stefano V. Albrecht
~~24~~ 28 January, 2025

Upcoming Event

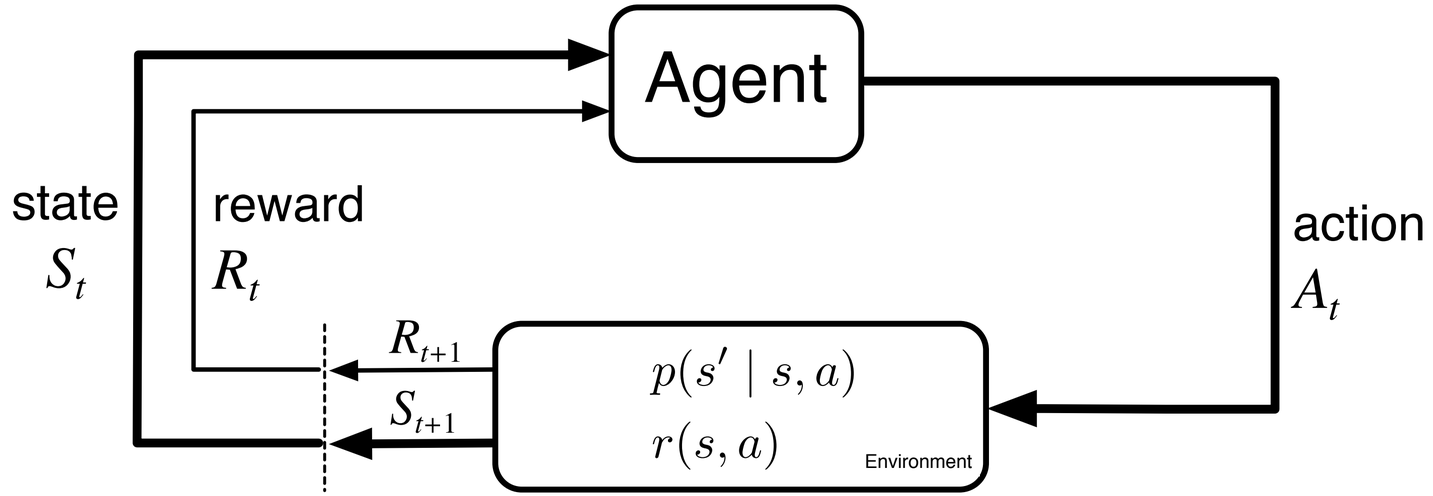
Lecture Outline

1. MDPs: Recap & Part 2
2. Main quantities: Policies, Values.
3. Algorithms for Solving MDPs: Value Iteration, Policy Iteration.

The Reinforcement Learning Loop



The RL Loop with an MDP



Markov Decision Processes

$\mathcal{S}$ Set of states

Markov Decision Processes

$\mathcal{S}$ Set of states

$\mathcal{A}$ Set of actions

`king's bishop to a6`

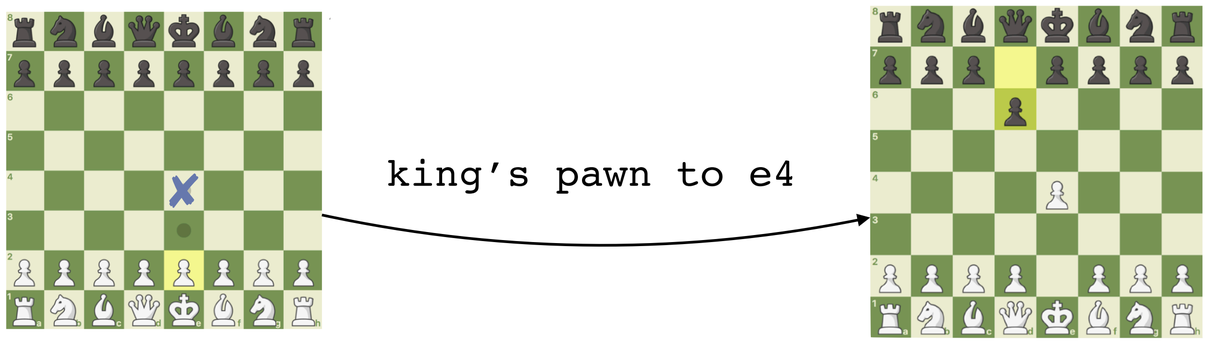
`king's pawn to e4`

Markov Decision Processes

$\mathcal{S}$ Set of states

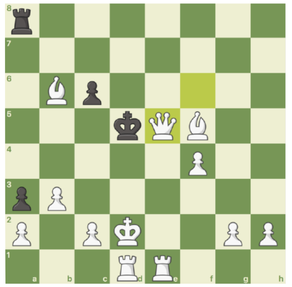
$\mathcal{A}$ Set of actions

$p(s' \mid s, a)$ Transition function



Markov Decision Processes

$\mathcal{S}$	Set of states
$\mathcal{A}$	Set of actions
$p(s' \mid s, a)$	Transition function
$r(s, a)$	Reward function

Win!  **+1**

 **-1** *Loss!*

Markov Decision Processes

$\mathcal{S}$	Set of states
$\mathcal{A}$	Set of actions
$p(s' \mid s, a)$	Transition function
$r(s, a)$	Reward function

Sometimes will write:

$(\mathcal{S}, \mathcal{A}, p)$ $\quad$ $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$

$(\mathcal{S}, \mathcal{A}, p, r)$ $\quad$ $(\mathcal{S}, \mathcal{A}, p, r, \gamma, s_0)$

Main Quantities

Given an MDP, $(\mathcal{S}, \mathcal{A}, p, r)$, interested in:

- Policy: $\pi(a \mid s)$

Main Quantities

Given an MDP, $(\mathcal{S}, \mathcal{A}, p, r)$, interested in:

- Policy: $\pi(a \mid s)$
- Discount factor: $\gamma \in [0, 1)$

Main Quantities

Given an MDP, $(\mathcal{S}, \mathcal{A}, p, r)$, interested in:

- Policy: $\pi(a \mid s)$
- Discount factor: $\gamma \in [0, 1)$
- Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$

Main Quantities

Given an MDP, $(\mathcal{S}, \mathcal{A}, p, r)$, interested in:

- Policy: $\pi(a \mid s)$
- Discount factor: $\gamma \in [0, 1)$
- Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$
- State-value: $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$

Main Quantities

Given an MDP, $(\mathcal{S}, \mathcal{A}, p, r)$, interested in:

- Policy: $\pi(a \mid s)$
- Discount factor: $\gamma \in [0, 1)$
- Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$
- State-value: $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
- Action-value $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$

The Bellman Equation: State-Value

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v_\pi(s')$$

Immediate reward

discounted

expected

future value

The Bellman Equation: State-Value

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) r(s,a) + \gamma \, \mathbb{E}_{s'}[v_\pi(s')]$$

Immediate reward

discounted

expected future value

The Bellman Equation: Action-Value

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \cdot v_\pi(s')$$

Immediate reward | discounted | expected | future value

The Bellman Equation: Action-Value

$$q_\pi(s,a) = r(s,a) + \gamma \, \mathbb{E}_{s'}[v_\pi(s')]$$

Immediate reward

discounted

expected future value

Optimality

Policy π is **optimal** if:

$$v_\pi(s) = v_*(s) = \max_{\pi'} v_{\pi'}(s)$$

$$q_\pi(s, a) = q_*(s, a) = \max_{\pi'} q_{\pi'}(s, a)$$

Optimality

Policy π is **optimal** if:

$$v_\pi(s) = v_*(s) = \max_{\pi'} v_{\pi'}(s)$$

$$q_\pi(s,a) = q_*(s,a) = \max_{\pi'} q_{\pi'}(s,a)$$

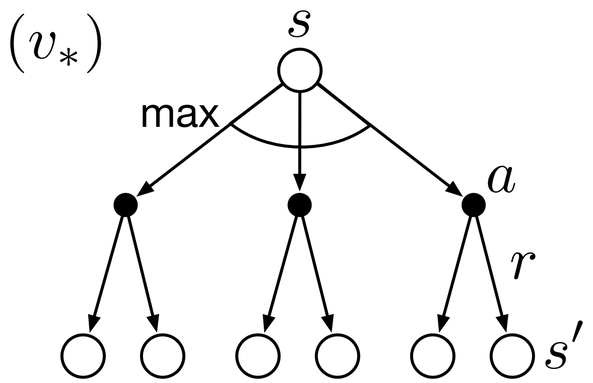
By the Bellman Equation, this implies that for any optimal policy:

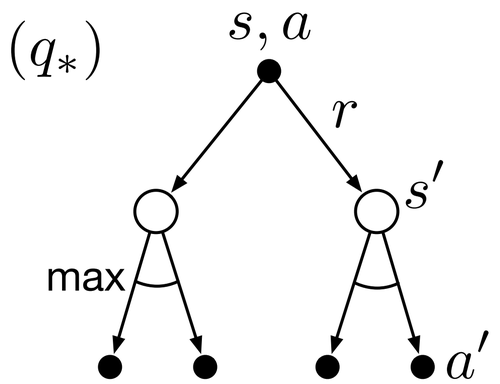
$$\forall \hat{\pi} \forall s : v_\pi(s) \geq v_{\hat{\pi}}(s)$$

Optimal Value Functions, Policies

$$v_*(s) = \max_{a \in \mathcal{A}} \left(\sum_{s', r} p(s', r | s, a) \left[r + \gamma \, v_*(s') \right] \right)$$

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma \; v_*(s')]$$

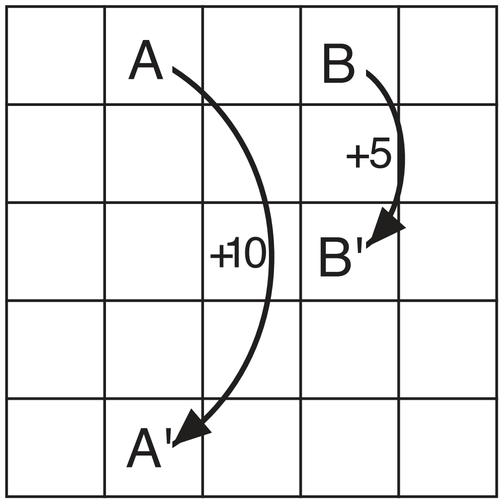




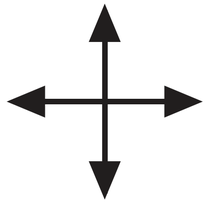
Example: Gridworld

Gridworld:

- States: cell locations in a grid
- Actions: Move north/south/east/west.
- Rewards: -1 off grid, +10/+5 in A/B, 0 otherwise.



$$\pi(a \mid s) \sim \text{Unif}(\mathcal{A})$$

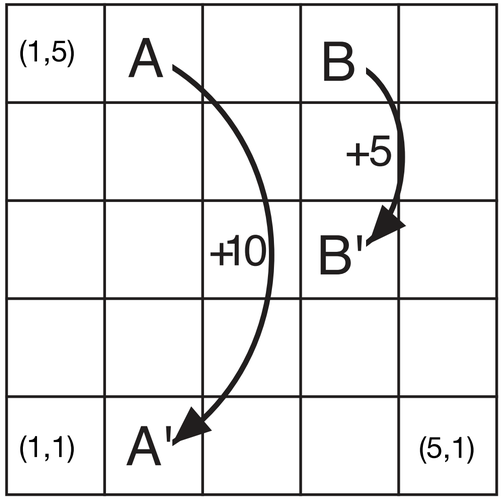


Actions

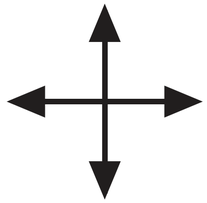
Discussion

Gridworld:

- States: cell locations in a grid
- Actions: Move north/south/east/west.
- Rewards: -1 off grid, +10/+5 in A/B, 0 otherwise.



$$\pi(a \mid s) \sim \text{Unif}(\mathcal{A})$$



Actions

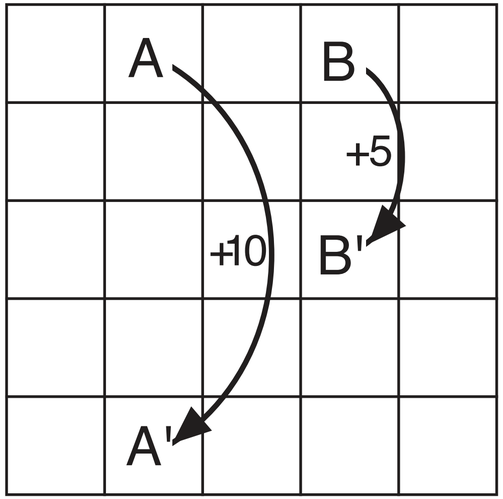
Discussion: What is the lowest value state? The highest?

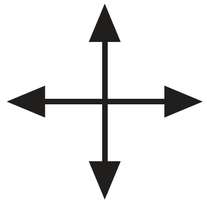
What are their values, roughly?

Example: Gridworld

Gridworld:

- States: cell locations in a grid
- Actions: Move north/south/east/west.
- Rewards: -1 off grid, +10/+5 in A/B, 0 otherwise.





Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

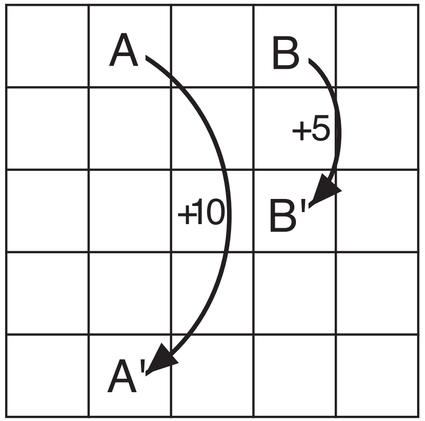
$$\pi(a \mid s) \sim \text{Unif}(\mathcal{A})$$

$$v_\pi(s)$$

Example: Gridworld — Optimal Policy

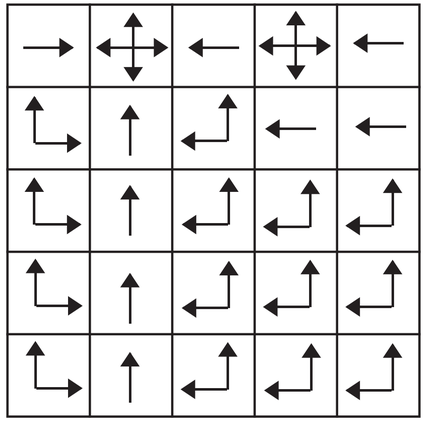
Gridworld:

- States: cell locations in a grid
- Actions: Move north/south/east/west.
- Rewards: -1 off grid, +10/+5 in A/B, 0 otherwise.



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

$v_{\pi_*}(s)$



$\pi_*(s)$

Value: System of Linear Equations

Bellman Equations form a system of $n = |\mathcal{S}|$ linear equations

$$
\begin{aligned}
v_\pi(s_1) &= \sum_a \pi(a|s_1) \sum_{s',r} p(s', r|s_1, a) \left[r + \gamma v_\pi(s')\right] \\
v_\pi(s_2) &= \sum_a \pi(a|s_2) \sum_{s',r} p(s', r|s_2, a) \left[r + \gamma v_\pi(s')\right] \\
&\vdots \\
v_\pi(s_n) &= \sum_a \pi(a|s_n) \sum_{s',r} p(s', r|s_n, a) \left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$

Value function is unique solution — solve using any method to solve linear system.

Recap: The Main Ideas

- **Markov Decision Process:** the canonical way to model RL problems
- **Policy:** π is a strategy for assigning actions to states.
- **Value:** $v_\pi(s)$, **Action-value:** $q_\pi(s, a)$ capture expected cumulative discounted reward

  —> *Long term* view of the quality of a policy.
- **Goal:** Find a policy that maximises *value.*

Two Problems: RL, and Planning

Definition: The RL Problem (MDPs)

Given: $\mathcal{S}, \mathcal{A}$, query access to p

Repeat, for t = 0, 1, ...

1. Agent selects action $A_t \in \mathcal{A}$
2. Agent observes

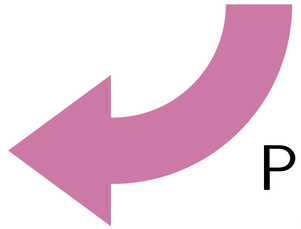
$$S_t, R_t \sim p(s', r \mid s, a)$$

Goal: maximise total reward.

Definition: Planning (MDPs)

Given: an MDP, $(\mathcal{S}, \mathcal{A}, p)$

Output: An optimal policy, π



Planning algorithms can inspire RL algorithms

Reinforcement Learning

Dynamic Programming: Part 1

David Abel, Michael Herrmann
Based on slides by Stefano V. Albrecht
~~24~~ 28 January, 2025

Lecture Outline

1. Policy Iteration
   (a) Iterative policy evaluation
   (b) Policy Improvement
2. Value Iteration
3. Asynchronous and generalised DP

Planning: Solving an MDP

Definition: Planning (MDPs)

Given: an MDP, $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$

Output: An optimal policy, π_*

$$\pi_* = \arg\max_{\pi} v_\pi(s), \ \ \forall s$$

Dynamic Programming

Dynamic programming is a *family of algorithms*

—> Main idea: use Bellman Equations to organise search for good policies:

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

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$$

DP Algorithm 1: Policy Iteration

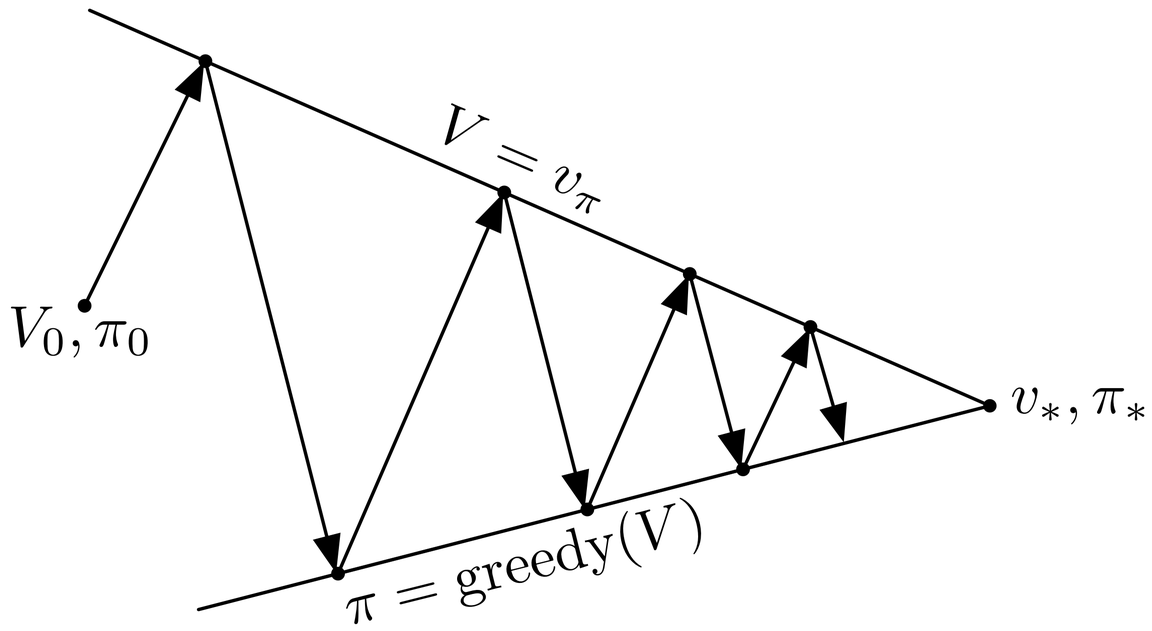
First algorithm: **Policy Iteration**, consists of two phases:

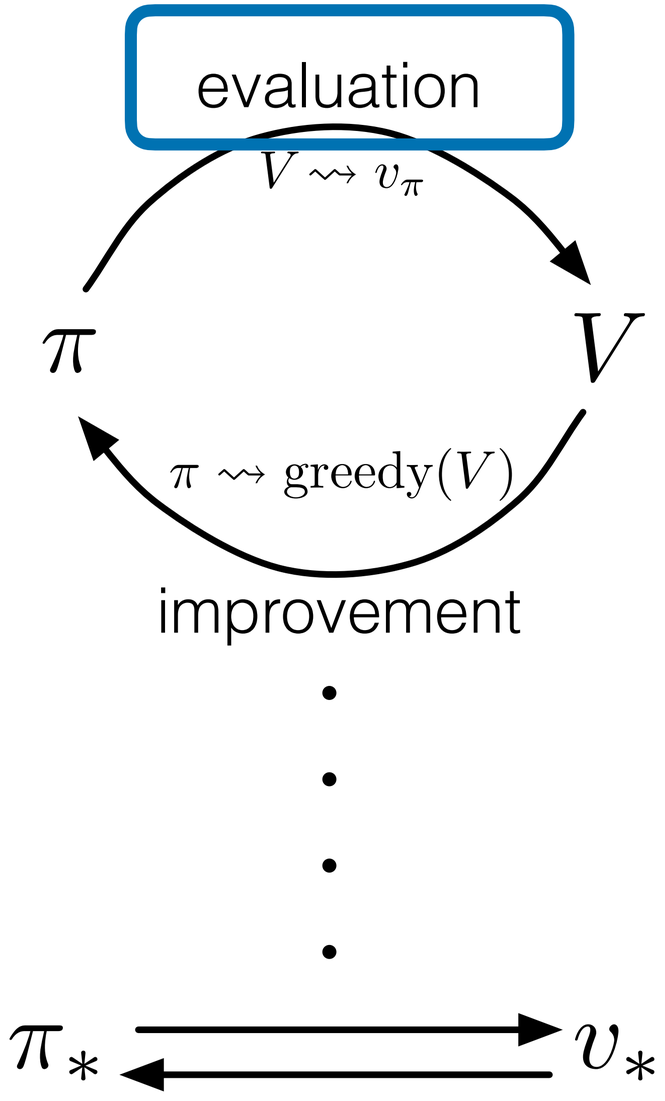
1. (E) Policy evaluation: compute v_π for π
2. (I) Policy improvement: make policy π greedy w.r.t. v_π

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

Process converges to optimal policy!

DP Algorithm 1: Policy Iteration

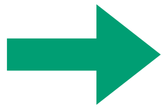




E: Policy Evaluation

Bellman Equations form a system of $n = |\mathcal{S}|$ linear equations

$$
\begin{aligned}
v_\pi(s_1) &= \sum_a \pi(a|s_1) \sum_{s',r} p(s', r|s_1, a) \left[r + \gamma v_\pi(s')\right] \\
v_\pi(s_2) &= \sum_a \pi(a|s_2) \sum_{s',r} p(s', r|s_2, a) \left[r + \gamma v_\pi(s')\right] \\
&\vdots \\
v_\pi(s_n) &= \sum_a \pi(a|s_n) \sum_{s',r} p(s', r|s_n, a) \left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$



Gauss elimination has complexity $\mathcal{O}(n^3)$

Iterative Policy **Evaluation**

Algorithm Sketch:

- Initialise $v_0(s) = 0$
- Update $v_k \longrightarrow v_{k+1}$ for each state s

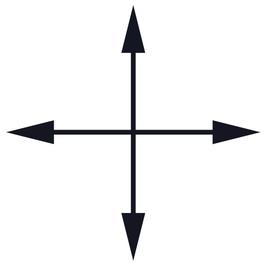
$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right]$$

Note: Sequence converges to v_π, stop when changes no more changes to v_k

Iterative Policy **Evaluation**: Pseudo-code

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Example: Gridworld



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$$R_t = -1$$
on all transitions

States: cell location in grid; grey squares are terminal

Actions: move north, south, east, west

Rewards: -1 until terminal state reached (recall: absorbing state, reward 0)

Undiscounted: $\gamma = 1$

Example: Policy Evaluation in Gridworld

$$\pi(a \mid s) \sim \text{Unif}(\mathcal{A})$$

$k = 0$

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

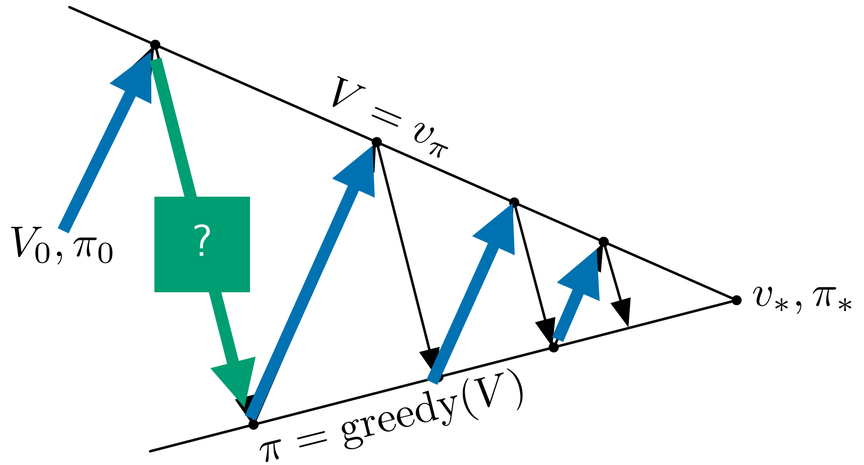
$k = \infty$

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

Policy Iteration: Two Steps

First algorithm: **Policy Iteration**, consists of two phases:

1. **(E) Policy evaluation**: compute v_π for π
2. **(I) Policy improvement**: make policy π greedy w.r.t. v_π

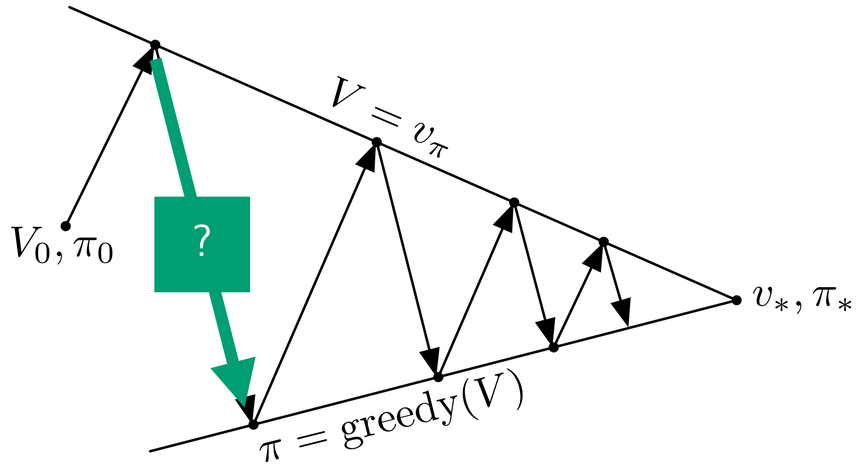


Policy Iteration

$$\forall s \in \mathcal{S}$$

$$\pi'(s) = \arg\max_a q_\pi(s,a)$$
$$= \arg\max_a \sum_{s',r} p(s',r \mid s,a)[r + \gamma v_\pi(s')]$$

2. **(I) Policy improvement**: make policy π greedy w.r.t. v_π



Policy Improvement Theorem

Policy Improvement Theorem

Let π and π' be policies such that for all s:

$$\sum_a \pi'(a|s)\, q_\pi(s,a) \;\geq\; \sum_a \pi(a|s)\, q_\pi(s,a)$$

$$= \; v_\pi(s)$$

Then π' must be as good as or better than π:

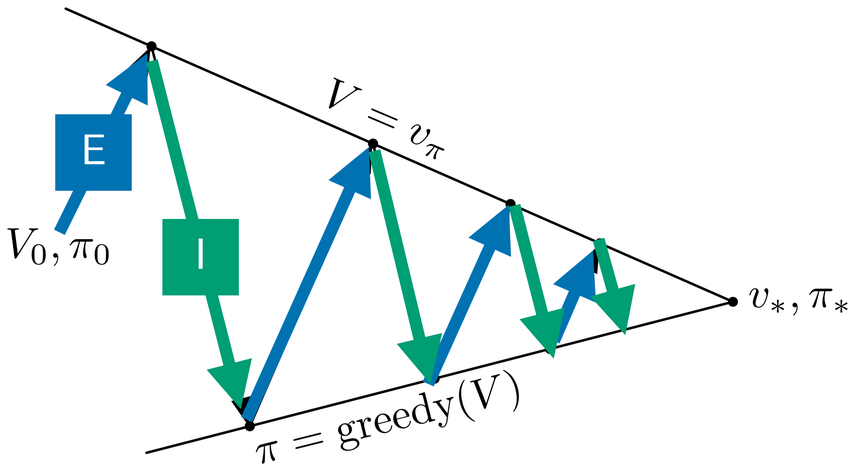
$$\forall s : v_{\pi'}(s) \geq v_\pi(s)$$

Policy Iteration

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $a \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ $false$
   If *policy-stable*, then stop and return V and π; else go to 2



DP Algorithm 2: Value Iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

DP Algorithms: Policy Iteration and Value Iteration

Policy Iteration

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      $a \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
      If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

Value Iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
      $v \leftarrow V(s)$
      $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
   $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Reading

- **RL book, Chapter 4 (4.1–4.7)**

  (Iterative Policy Evaluation proof not examined)

- Optional:
  - Dynamic Programming and Optimal Control by Dimitri P. Bertsekas

    http://www.athenasc.com/dpbook.html
  - On the Complexity of Solving Markov Decision Problems by Littman, Dean, Kaelbling

    https://arxiv.org/pdf/1302.4971

Policy Improvement Theorem: Proof Sketch

$$
\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) && \text{(here for deterministic policies)} \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)\right] \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right] \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s\right] && \text{(by premise)} \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma \mathbb{E}_{\pi'}\left[R_{t+2} + \gamma v_\pi(S_{t+2}) \mid S_{t+1}, A_{t+1} = \pi'(S_{t+1}) \mid S_t = s\right]\right] \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s\right] \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \mid S_t = s\right] \\
&\dots \\
&\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \mid S_t = s\right] \\
&= v_{\pi'}(s)
\end{aligned}
$$