《强化学习与控制》
--
Stochastic DP

Shengbo Eben Li
(李升波)

Intelligent Driving Lab (iDLab)

Tsinghua University

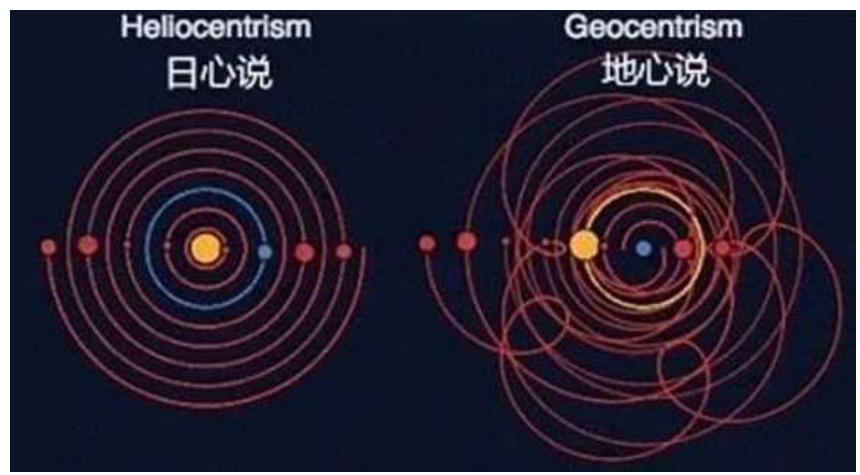
Fear Is Fear Itself

Nothing in life is to be feared,
it is only to be understood.
Now is the time to understand more,
so that we may fear less.

-- Marie Curie (1867 - 1934)

William of Occam (~ 14 century)

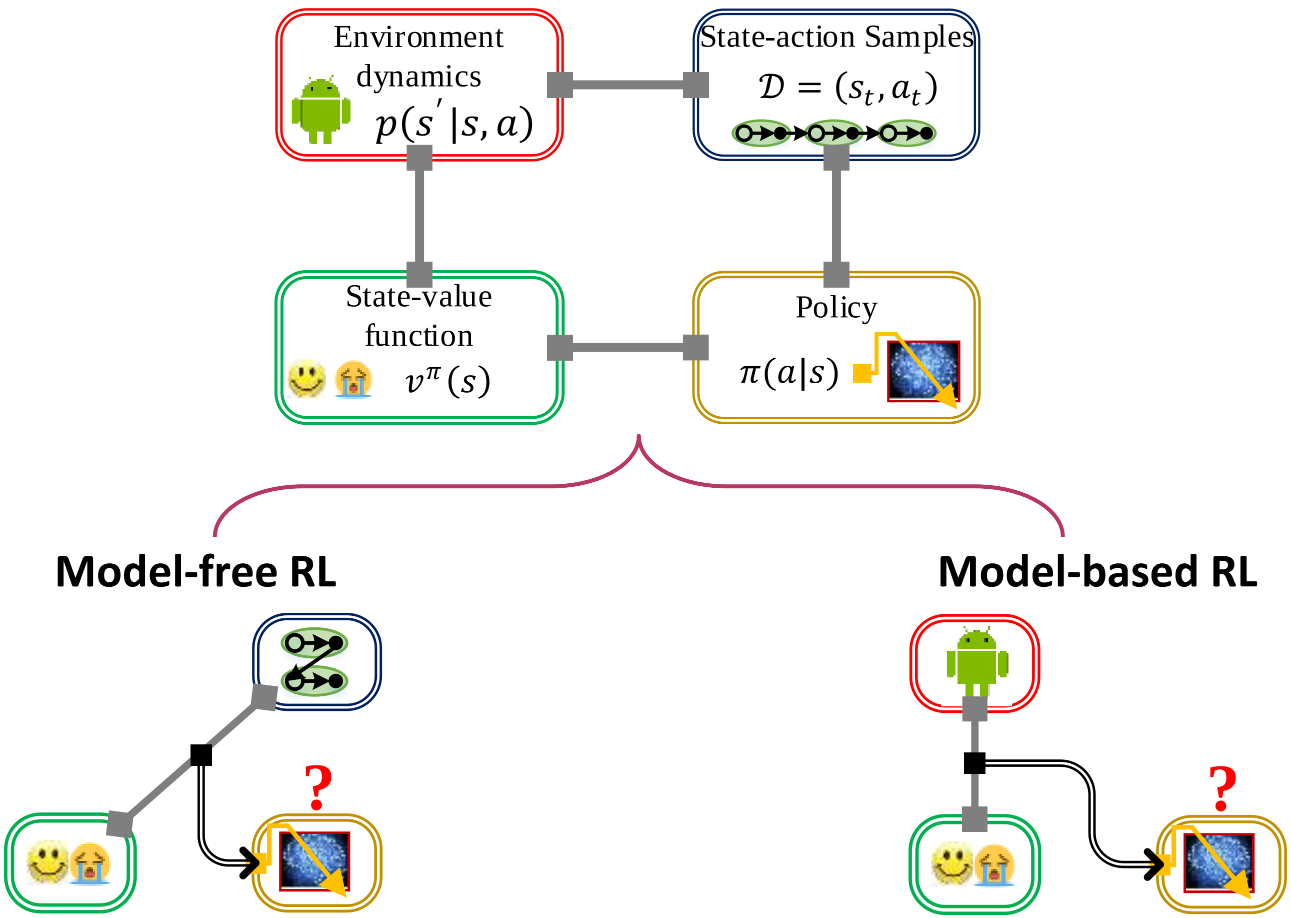
Plurality ought never be posited without necessity



Outline

1. Intro to Stochastic DP
2. DP Policy Iteration
3. DP Value Iteration
4. A Unified Framework

Classification of RLs



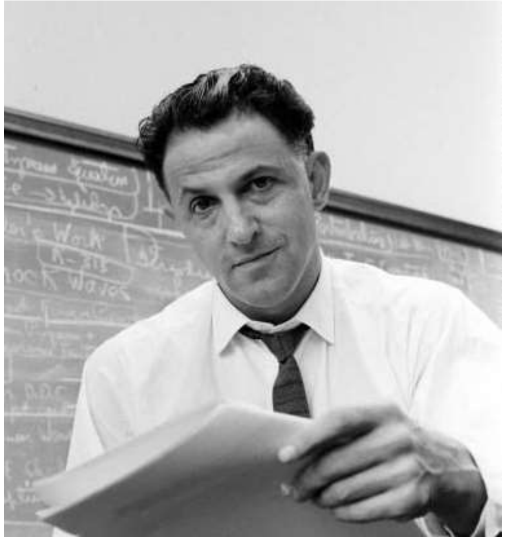
Dynamic Programming (DP)

- The third pillar of optimal control, like Calculus of Variations and Pontryagin's Maximum Principle

First developed by Richard Bellman

- Worked in Rand Corporation
- Dynamic programming (1953)
- > 600 papers, 35 books & 7 monograph

Awarded the IEEE Medal of Honor in 1979, "for contributions to decision processes and control system theory, particularly the creation and application of **dynamic programming**"



Richard Bellman
(1920-1984)

Introduction to DP

- Algorithms that use dynamic programming (DP)
  - Cocke-Younger-Kasami algorithm
  - Knuth's word wrapping algorithm
  - Viterbi algorithm
  - Earley algorithm
  - Needleman-Wunsch algorithm
  - Floyd's all-pairs shortest path algorithm
  - Dynamic time warping algorithm
  - Selinger algorithm
  - De Boor algorithm
  - Duckworth-Lewis algorithm
  - Recursive least squares algorithm
  - Bellman-Ford algorithm
  - Kadane's algorithm

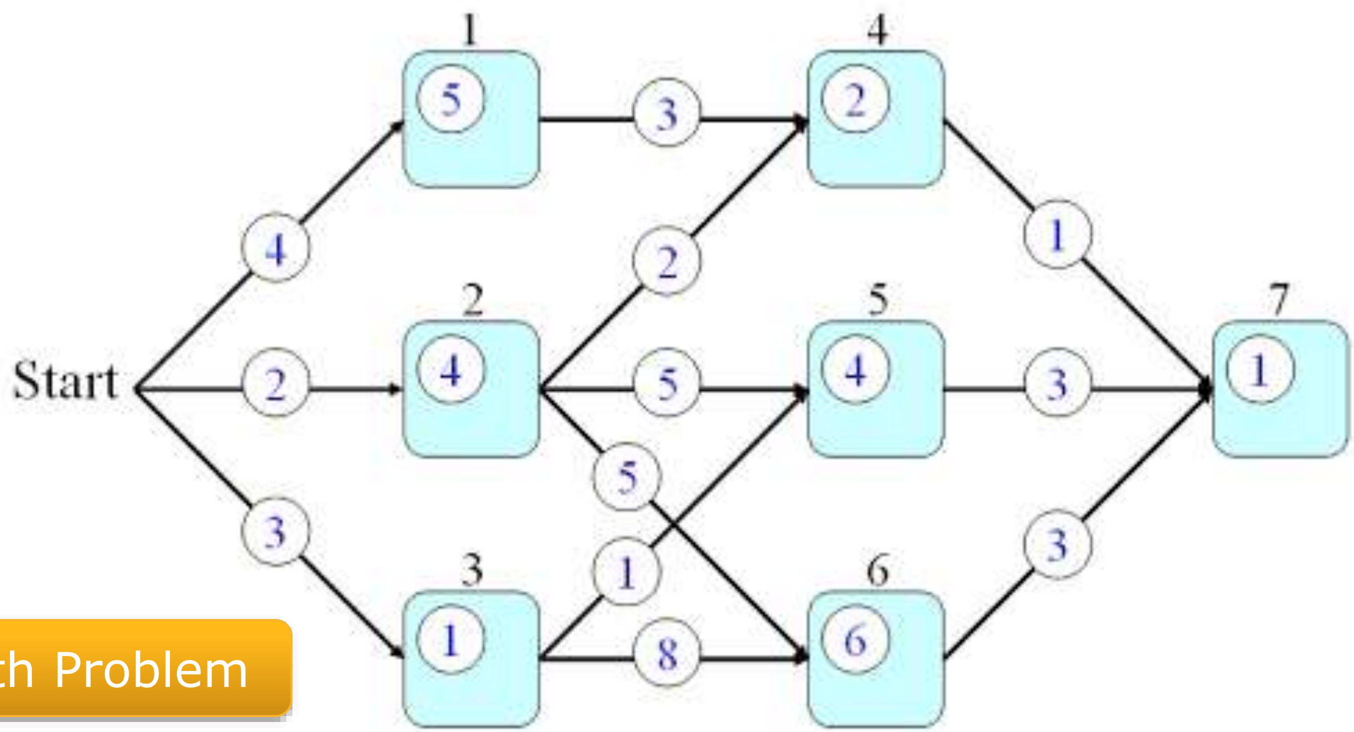
How does “DP” come from?

- *Eye of the Hurricane: An Autobiography* (1984)
  - I spent the Fall quarter (of 1950) at RAND. My first task was to find a name for multistage decision processes.
  - The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word "research". He would get violent if people used the term research in his presence.
  - The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation.
  - Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.

Bellman's Principle of Optimality

◻ Intuitive Explanation

Tail of an optimal policy remains to be the optimal policy for a subproblem induced by applying the first action of optimal policy



Shortest Path Problem

Stochastic Control Systems

Consider stochastic state space model

$$s_{t+1} = f(s_t, a_t, \xi_t)$$

- ξ_t is a random noise with known distribution, i.e., i.i.d., and independent of initial state s_0

Markov property

$$p(s_{t+1}|s_t, \dots, s_2, s_1, s_0) = p(f(s_t, a_t, \xi_t)|s_t, \dots, s_2, s_1, s_0)$$

- Since ξ_t is independent of s_0 and, $a_0, \cdots, a_t$ are arbitrary variables

$$\begin{aligned}
&= p(f(s_t, a_t, \xi_t)|s_t, \dots, s_2, s_1) \\
&= p\big(f(s_t, a_t, \xi_t)|s_t, \dots, s_2, f(s_0, a_0, \xi_0)\big) \\
&= p(f(s_t, a_t, \xi_t)|s_t, \dots, s_3, s_2) \\
&= p\big(f(s_t, a_t, \xi_t)|s_t, \dots, s_3, f(f(s_0, a_0, \xi_0), a_1, \xi_1)\big) \\
&= p(f(s_t, a_t, \xi_t)|s_t, \dots, s_3)
\end{aligned}$$

- Roll forward until s_t at time t

$$= p(f(s_t, a_t, \xi_t)|s_t) = p(s_{t+1}|s_t)$$

Two kinds of objective function

(1) Average Cost

- Equal to average return

$$G_{\text{avg}}(\pi) = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left\{\sum_{i=0}^{T-1} r_{t+i}\right\}$$

- **VS** discounted return weighted by stationary state distribution

$$J_{\text{avg}}(\pi) = \sum d_{\pi}(s) v_{\gamma}^{\pi}(s)$$

Policy-dependent state distribution

(2) Discounted Cost

- Weighted expectation of discounted return

$$J_{\gamma}(\pi) = \sum d_{\text{avg}}^{*}(s) v_{\gamma}^{\pi}(s)$$

Given initial state distribution Independent of any policy

- Define two optimal policies

$$\pi_{\text{avg}}^{*} = \arg\max_{\pi} J_{\text{avg}}(\pi) \qquad \pi_{\gamma}^{*} = \arg\max_{\pi} J_{\gamma}(\pi)$$

Two kinds of objective function

Relation between average cost & discounted cost

- Theorem: For an arbitrary policy π, we have

$$J_{\text{avg}}(\pi) = \frac{1}{1-\gamma} G_{\text{avg}}(\pi)$$

Cost equivalence theorem

$$\max_{\pi} J_{\text{avg}}(\pi) \Leftrightarrow \max_{\pi} G_{\text{avg}}(\pi)$$

- Theorem: For two optimal policies π^*_{avg} and π^*_{γ}, their performance measures satisfy

Self-optimality

$$J_{\text{avg}}(\pi^*_{\text{avg}}) = \underbrace{\sum d^*_{\text{avg}}(s)}_{\text{Definition for average cost}} \underbrace{v_{\gamma}^{\pi^*_{\text{avg}}}(s)}_{\text{Definition for discounted cost}} = J_{\gamma}(\pi^*_{\text{avg}}) \leq J_{\gamma}(\pi^*_{\gamma})$$

Definition for average cost

Definition for discounted cost

Two kinds of objective function

The inequality from Tsinghua's iDLab

$$\frac{1}{1-\gamma}G_{\text{avg}}(\pi_\gamma^*) \underbrace{\leq}_{\text{Self-optimality}} \frac{1}{1-\gamma}G_{\text{avg}}(\pi_{\text{avg}}^*) \underbrace{=}_{\text{Cost equivalence}} J_\gamma(\pi_{\text{avg}}^*) \underbrace{\leq}_{\text{Self-optimality}} J_\gamma(\pi_\gamma^*)$$

- (1) Cost equivalence: (1) Definition of average cost is equal to that of discounted cost under the policy π_{avg}^*; (2) Cross-measures of policy π_{avg}^* under discounted cost and average cost have a linear relationship

- (2) Self-optimality: (1) π_{avg}^* is no worse than any arbitrary policy under average cost $G_{\text{avg}}(\cdot)$; (2) π_γ^* is optimal among all policies under discounted cost $J_\gamma(\cdot)$

Two kinds of objective function

Example: Stochastic LQ control (w/ maximizer)

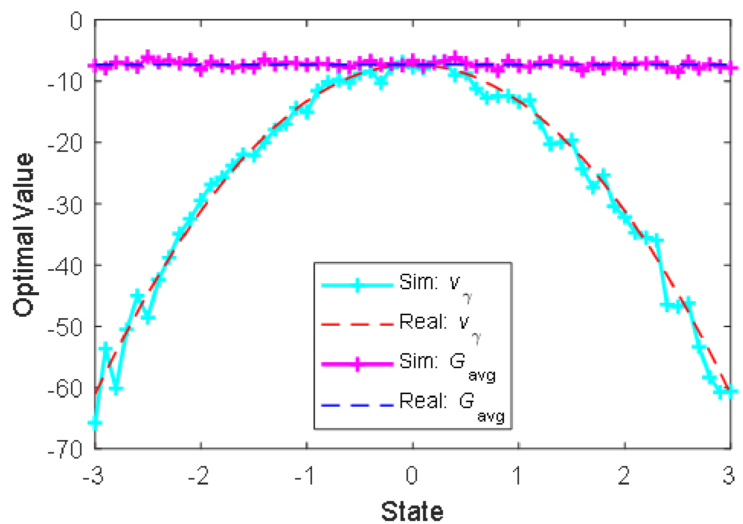
$$s_{t+1} = As_t + Ba_t + \xi_t, \xi_t \sim \mathcal{N}(0, \sigma^2)$$

- Discounted cost

$$\max_{a_t, a_{t+1}, \dots, a_\infty} J = \sum_{i=0}^{\infty} \gamma^i \left(s_{t+i}{}^{\mathrm{T}} Q s_{t+i} + a_{t+i}{}^{\mathrm{T}} R a_{t+i} \right)$$

- Average cost

$$\max_{a_t, a_{t+1}, \dots, a_\infty} G = \lim_{T \to \infty} \frac{1}{T} \sum_{i=0}^{T-1} \left(s_{t+i}{}^{\mathrm{T}} Q s_{t+i} + a_{t+i}{}^{\mathrm{T}} R a_{t+i} \right)$$

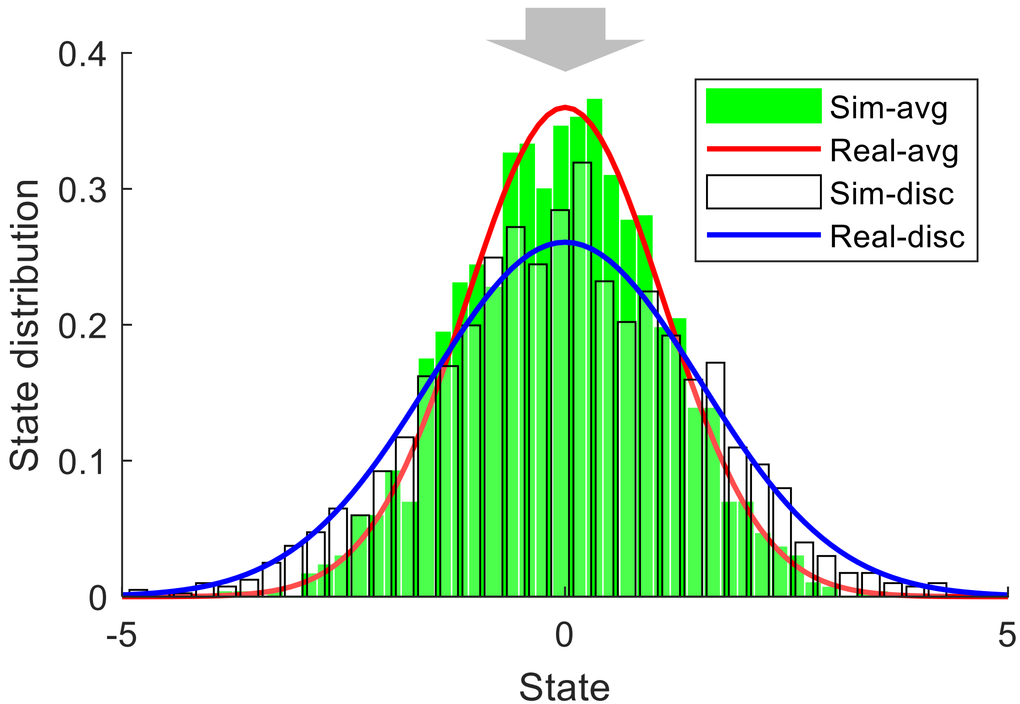


Two kinds of objective function

Example: Stochastic LQ control

$$d_{\pi^*_{\text{avg}}}(s) = \sqrt{1-\left(A-BK_{\text{Avg}}\right)^2}\,\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{s^2}{2\sigma^2}\left(1-\left(A-BK_{\text{Avg}}\right)^2\right)\right)$$

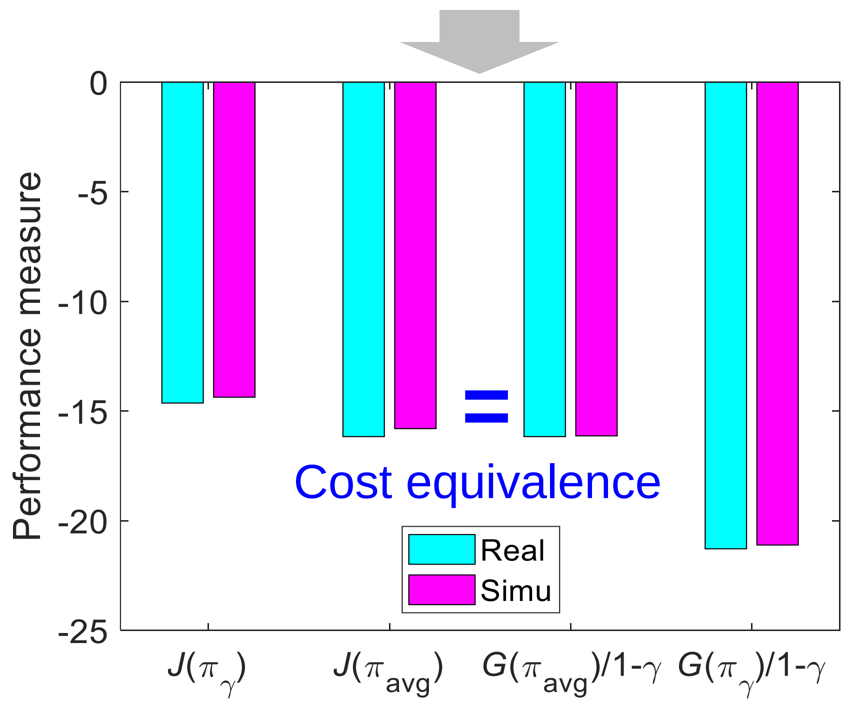
$$d_{\pi^*_{\gamma}}(s) = \sqrt{1-\left(A-BK_{\gamma}\right)^2}\,\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{s^2}{2\sigma^2}\left(1-\left(A-BK_{\gamma}\right)^2\right)\right)$$



Two kinds of objective function

Example: Stochastic LQ control

Performance measure	Real	Simulation	Error
$J_\gamma(\pi_\gamma^*)$	-14.64	-14.38	1.9%
$J_\gamma(\pi_{\text{avg}}^*)$	-16.17	-16.00	1.1%
$(1-\gamma)^{-1}G_{\text{avg}}(\pi_{\text{avg}}^*)$	-16.17	-16.41	1.5%
$(1-\gamma)^{-1}G_{\text{avg}}(\pi_\gamma^*)$	-21.28	-21.22	0.2%



Stochastic Dynamic Programming (SDP)

Probabilistic model

$$\mathcal{P}_{ss'}^{a} \overset{\text{def}}{=} p(s'|s,a) = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

t : current time

s : state

a : action

Goal

- Maximize state-value function (w/o initial state distribution)

$$\pi^* = \arg\max_{\pi} \; v^{\pi}(s)\,, \forall s \in \mathcal{S}$$

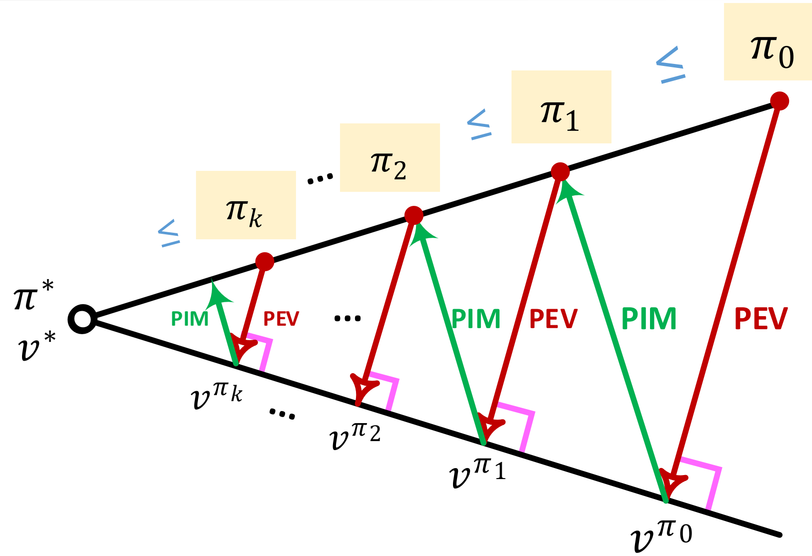
- Bellman equation of the first kind

$$v^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{a} \left(r_{ss'}^{a} + \gamma v^*(s') \right)$$

- Bellman equation of the second kind

$$q^*(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{a} \left(r_{ss'}^{a} + \gamma \max_{a' \in \mathcal{A}} q^*(s', a') \right)$$

How to solve Bellman equation?

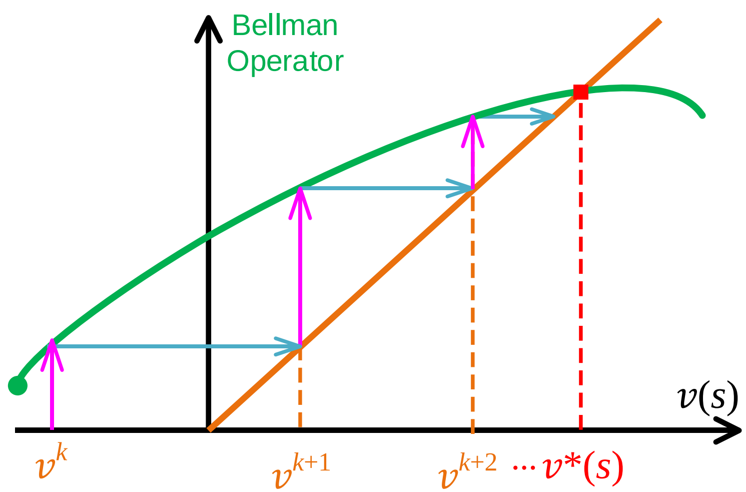


(1) Policy iteration

- PEV + PIM
- Search for a better policy

(2) Value iteration

- Fixed-point iteration to solve an equation



$$v^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \left(r_{ss'}^a + \gamma v^*(s') \right)$$

Fixed-point iteration

Outline

1. Intro to Stochastic DP
2. DP Policy Iteration
3. DP Value Iteration
4. A Unified Framework

Policy Iteration Algorithm

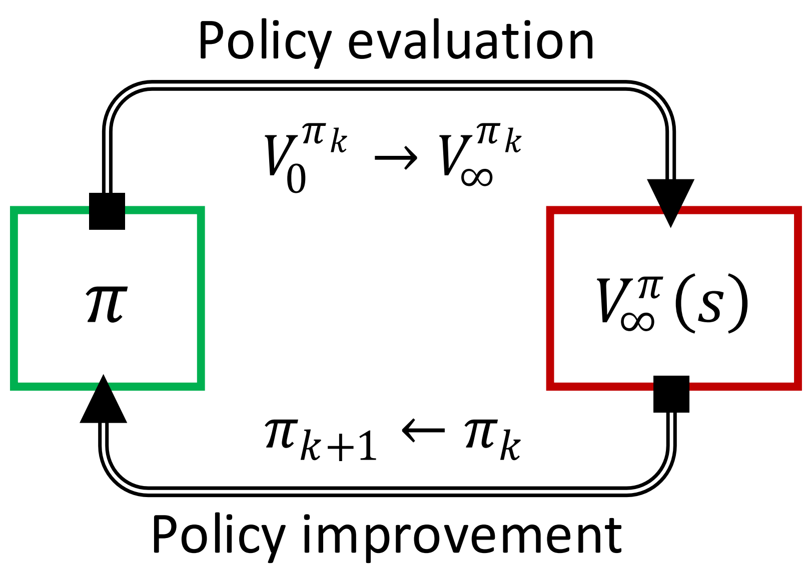
Two-step cyclic framework

- (1) Policy evaluation (**PEV**)

  Find corresponding "true" value function for a given policy π

- (2) Policy improvement (**PIM**)

  Find a better policy according to "true" value function $V_\infty^\pi(s)$



DP Policy Evaluation

- **Use self-consistency condition to calculate state-value function**

$$v^{\pi}(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left\{ \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^{a} \left(r_{ss'}^{a} + \gamma v^{\pi}(s') \right) \right\}, \forall s \in \mathcal{S}$$

- Three elements are known: (1) environment model $\mathcal{P}_{ss'}^{a}$, (2) policy $\pi(a|s)$ and (3) reward signal $r_{ss'}^{a}$
- Build a group of linear equations with $v^{\pi}(s), \forall s \in \mathcal{S}$ as the unknown variable to be solved

Iterative PEV algorithm

- Repeat j until to infinity

$$V_{j+1}^{\pi}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s)\left\{\sum_{s'\in\mathcal{S}} \mathcal{P}\left(r+\gamma V_j^{\pi}(s')\right)\right\}, \forall s\in\mathcal{S}$$

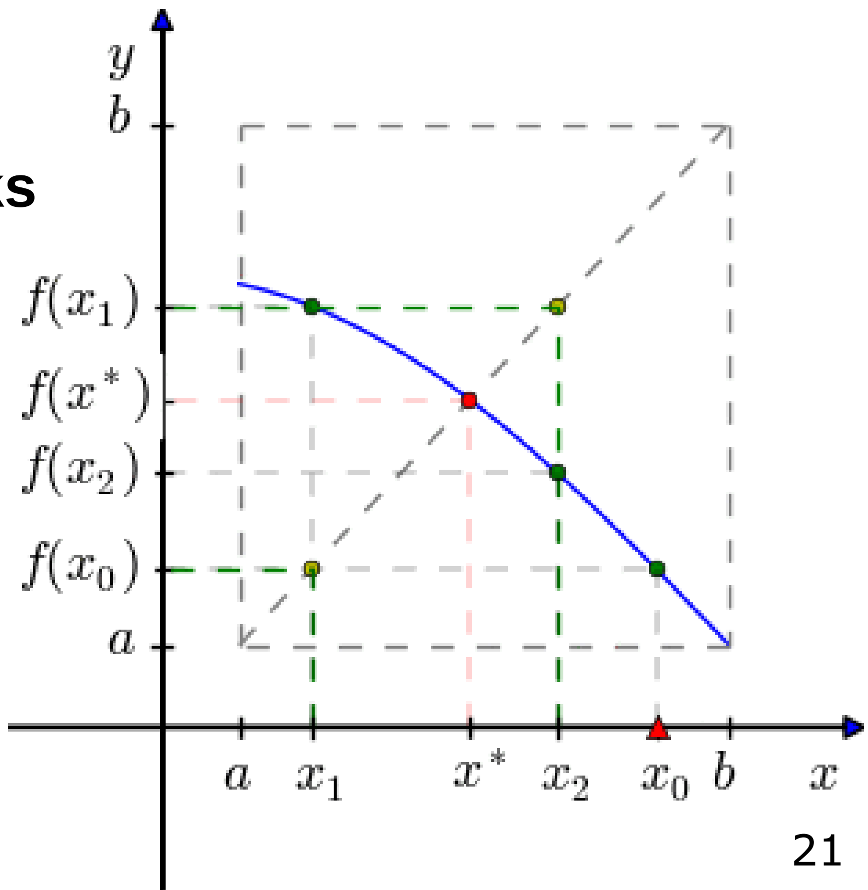
- End

How fixed-point iteration works

$$x = f(x)$$

Picard iteration

$$x_{k+1} \leftarrow f(x_k)$$



Revisit Self-consistency Condition

$$V^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\left\{\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^{a}\left(r_{ss'}^{a} + \gamma V^\pi(s')\right)\right\}, \forall s \in \mathcal{S}$$

$$\mathcal{S} = \{s_{(1)}, s_{(2)}, \cdots, s_{(n)}\}$$

Linear Algebraic Equations

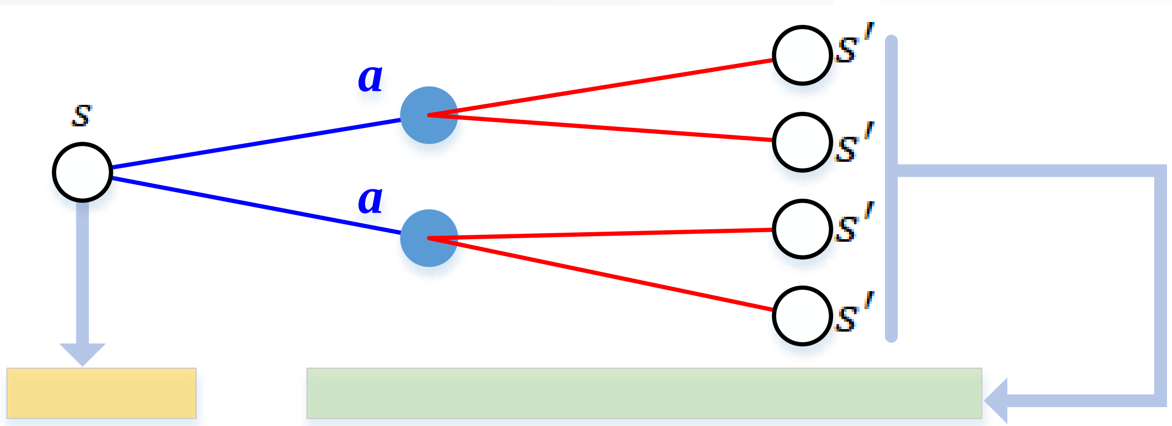
$$V^\pi(s_{(1)}) = \sum \pi \sum \mathcal{P}_{s_{(1)}s'_{(1)}}^{a}\left(r + \gamma V^\pi\left(s'_{(1)}\right)\right)$$

$$V^\pi(s_{(2)}) = \sum \pi \sum \mathcal{P}_{s_{(2)}s'_{(2)}}^{a}\left(r + \gamma V^\pi\left(s'_{(2)}\right)\right)$$

$$\cdots \ \cdots$$

$$V^\pi(s_{(n)}) = \sum \pi \sum \mathcal{P}_{s_{(n)}s'_{(n)}}^{a}\left(r + \gamma V^\pi\left(s'_{(n)}\right)\right)$$

Revisit Self-consistency Condition



$$\begin{bmatrix} V^\pi(s_{(1)}) \\ V^\pi(s_{(2)}) \\ \cdots \\ V^\pi(s_{(n)}) \end{bmatrix} = \gamma \begin{bmatrix} B_{11} & \cdots & B_{1n} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nn} \end{bmatrix} \times \begin{bmatrix} V^\pi(s_{(1)}) \\ V^\pi(s_{(2)}) \\ \cdots \\ V^\pi(s_{(n)}) \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Compact form

$$X = \gamma B X + b$$

$$X^\mathrm{T} = [V^\pi(s_{(1)}), V^\pi(s_{(2)}), \cdots, V^\pi(s_{(n)})],\ B = \{B_{ij}\}_{n\times n} \in \mathbb{R}^{n\times n}, b = \{b_i\}_{n\times 1} \in \mathbb{R}^n$$

Revisit Self-consistency Condition

Compact form

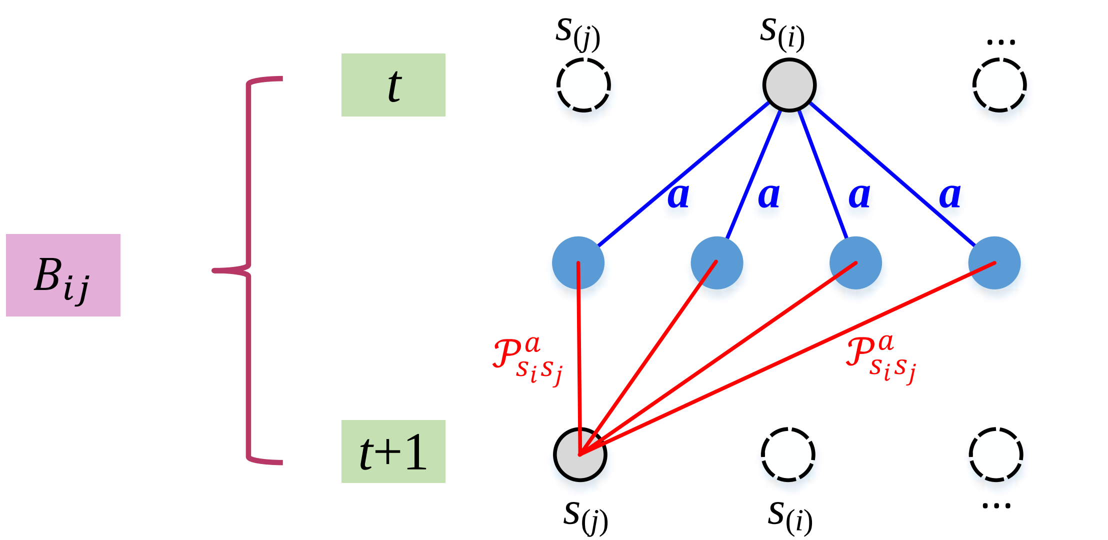
$$X = \gamma BX + b$$

where

$$B_{ij} = \sum_{a\in\mathcal{A}} \pi(a|s_{(i)})\mathcal{P}^a_{s_{(i)}s_{(j)}}$$

$$b_i = \sum_{a\in\mathcal{A}} \pi(a|s_{(i)}) \sum_j \mathcal{P}^a_{s_{(i)}s_{(j)}} r^a_{s_{(i)}s_{(j)}}$$

- Each element of B matrix



Revisit Self-consistency Condition

Short Remark:

- Self-consistency condition is linear : $X = \gamma BX + b$
- Can be directly solved if $A \stackrel{\text{def}}{=} I_{n\times n} - \gamma B$ is reversible

$$X = (I_{n\times n} - \gamma B)^{-1} b = A^{-1} b$$

- Theorem: if $0 < \gamma < 1$, A is reversible.
  - Proof:

$$\|B\|_\infty = \max \left| \sum_{j=1}^{n} B_{i,j} \right| = 1$$

$$\rho(\gamma B) \leq \|\gamma B\|_\infty = \gamma < 1$$

  - A unique solution exists for self-consistency condition
  - The complexity of calculating inverse matrix is $O(n^3)$

Convergence of Iterative PEV

Theorem: The operator in PEV is γ-contractive

$$\mathcal{L}(X) \stackrel{\text{def}}{=} \gamma B X + b$$

- Proof
$$\begin{aligned}
\left\|\mathcal{L}(X_{j+1}) - \mathcal{L}(X_j)\right\|_\infty &= \gamma \left\|B\left(X_{j+1} - X_j\right)\right\|_\infty \\
&\le \gamma \left\|B \max_{i\in\{1,2,\cdots,n\}}\left(\left|X_{j+1}^i - X_j^i\right|\right)\right\|_\infty \\
&= \gamma \left\|B \underbrace{\left\|X_{j+1} - X_j\right\|_\infty}_{\text{Inf-Norm = Scalar}}\right\|_\infty \\
&= \gamma \|B\|_\infty \left\|X_{j+1} - X_j\right\|_\infty \\
&= \gamma \left\|X_{j+1} - X_j\right\|_\infty
\end{aligned}$$

Note that $\|B\|_\infty = \max_i \sum_{j=1}^n B_{i,j} = 1$

Contraction mapping theorem

- The γ-contraction must have a unique fixed point
- $\mathcal{L}(X)$ converges at the linear rate of γ if X is in a closed space

DP Policy Improvement

- Greedy search w.r.t. estimated "true" state-value function

Stochastic policy

$$\pi'(a|s) \leftarrow \arg\max_{\pi'} \left\{ \sum \pi'(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}\left(r + \gamma V_{\infty}^{\pi}(s')\right) \right\}$$

Deterministic policy

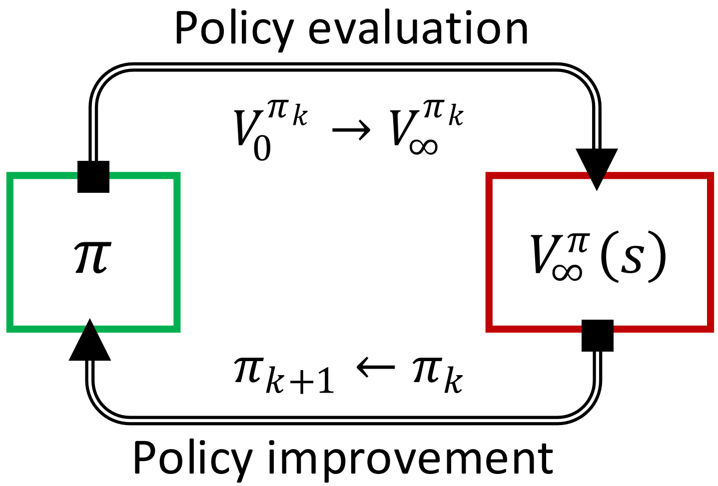
$$\pi'(s) \leftarrow \arg\max_{\pi'} \left\{ \sum_{s' \in \mathcal{S}} \mathcal{P}\left(r + \gamma V_{\infty}^{\pi}(s')\right) \right\}$$

- Greedy search satisfies element-by-element definition

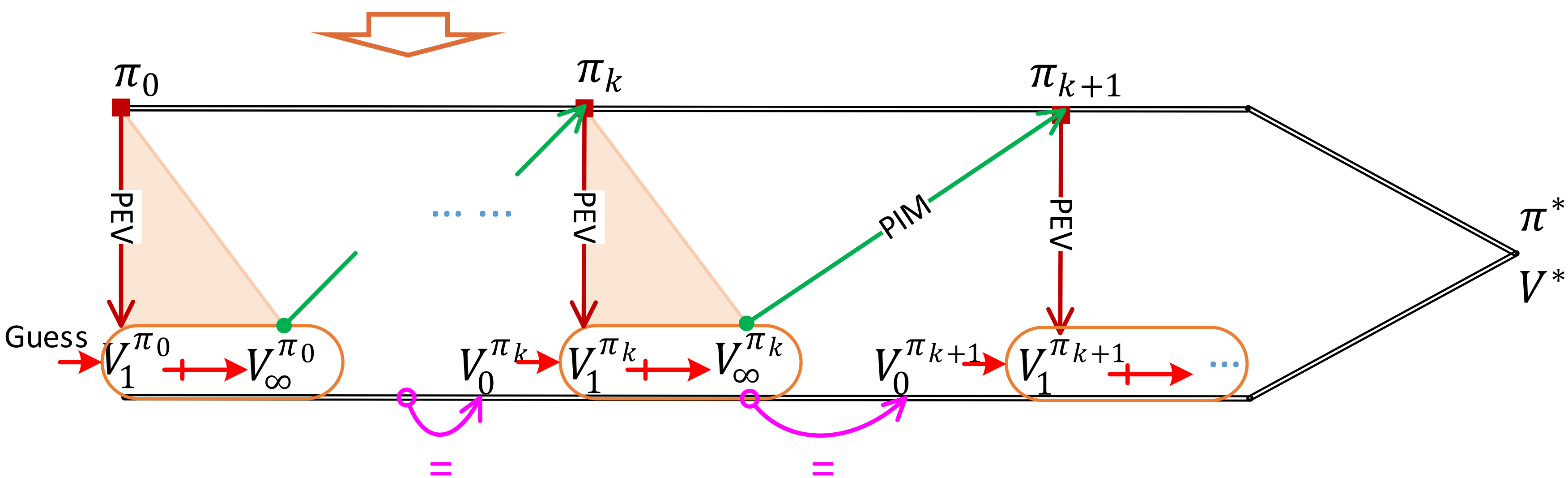
$$v^{\pi}(s) \leq v^{\pi'}(s), \forall s \in \mathcal{S}$$

Convergence of DP Policy Iteration

☐ Policy Evaluation (PEV) + Policy Improvement (PIM)



Convergence ?



Convergence of DP Policy Iteration

Proof

- The key is to prove that $V_\infty^{\pi_k}(s)$ is monotonically increasing

$$V_\infty^{\pi_0}(s) \le V_\infty^{\pi_1}(s) \le \cdots \le V_\infty^{\pi_k}(s) \le V_\infty^{\pi_{k+1}}(s) \le \cdots \le v^*$$

  - PIM: greedy search yields a better policy

$$v^{\pi_k}(s) \le v^{\pi_{k+1}}(s)$$

  - PEV: output true value of current policy

$$V_\infty^{\pi_k}(s) = v^{\pi_k}(s)$$

  - Replace v^{π_k} and $v^{\pi_{k+1}}$ with $V_\infty^{\pi_k}$ and $V_\infty^{\pi_{k+1}}$

$$V_\infty^{\pi_k}(s) \le V_\infty^{\pi_{k+1}}(s)$$

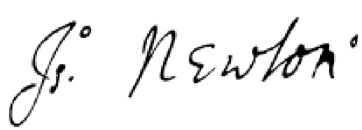
Bellman equation is satisfied!

- If PIM stops, i.e.,$\pi_\infty = \pi_{\infty+1}$

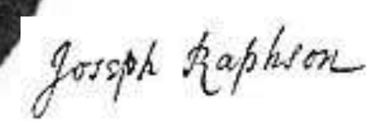
$$v^{\pi_\infty}(s) = v^{\pi_{\infty+1}}(s) = \max_a \sum_{s\prime\in\mathcal{S}} \mathcal{P}\big(r + \gamma v^{\pi_\infty}(s')\big), \forall s \in \mathcal{S}$$

- **Mechanism behind**
  - M Puterman & S Brumelle (1979)


Isaac Newton (1642 - 1726)

Joseph Raphson (1668 - 1712)

Explanation with Newton-Raphson Method

Newton–Raphson method

- Most widely used root-finding and second-order optimization
- Converge in a quadratic speed
- (1) as equation solver
- (2) as convex optimizer

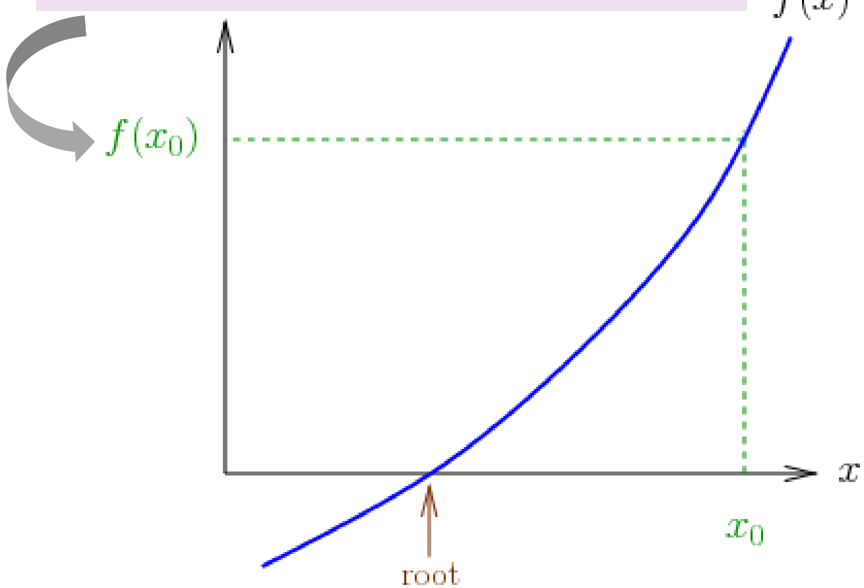
$$\min_{x} f(x)$$

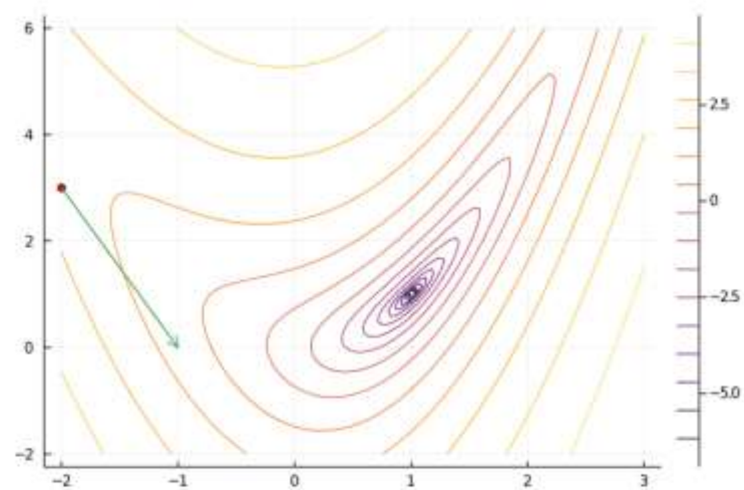
$$f(x) = 0$$

$$x_{k+1} = x_k - [\nabla f(x_k)]^{-1} f(x_k)$$

$$\nabla f(x) = 0$$

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$





Explanation with Newton-Raphson Method

MDP with 2 states and 2 actions

$$V^\pi(s_{(1)}) = \max_a \sum_{s'} \mathcal{P}^a_{s_{(1)}s'}\left(r + \gamma V^\pi(s')\right),$$

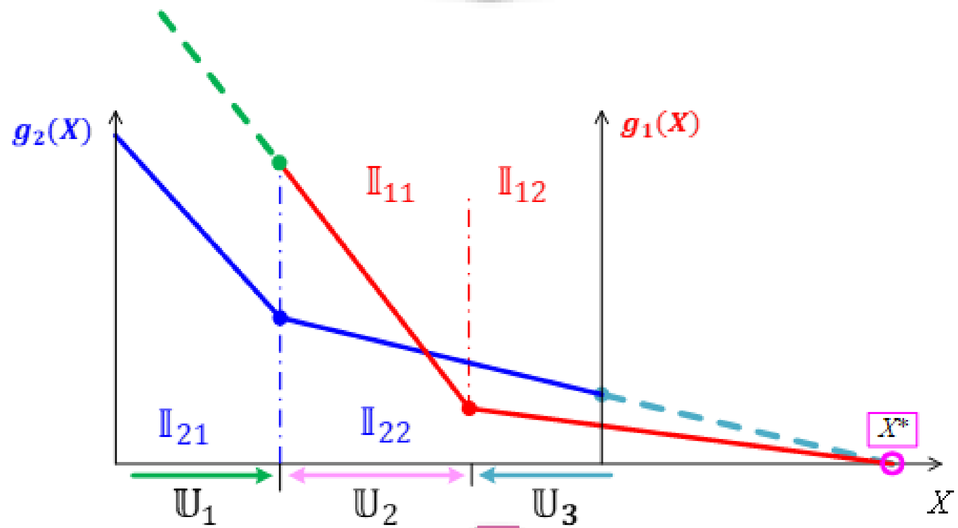
$$V^\pi(s_{(2)}) = \max_a \sum_{s'} \mathcal{P}^a_{s_{(2)}s'}\left(r + \gamma V^\pi(s')\right)$$

$a \in \{a_{(1)}, a_{(2)}\}$

$$g(X) = \mathcal{B}(X) - X,$$

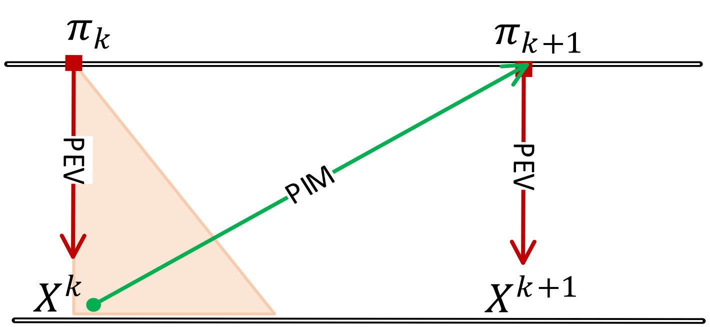
$$g_i(X) = \max_j \left(\gamma \beta_{ij}^{\mathrm{T}} X + \lambda_{ij}\right) - X_i\,, i, j \in \{1,2\}$$

Newton-Raphson Iteration



$$X_{k+1} = \gamma B_{\pi_{k+1}} X_{k+1} + b_{\pi_{k+1}}$$

DP Policy Iteration



$$X_{k+1} = \gamma B_{\pi_{k+1}} X_{k+1} + b_{\pi_{k+1}}$$

Outline

1. Intro to Stochastic DP
2. DP Policy Iteration
3. DP Value Iteration
4. A Unified Framework

Value Iteration Algorithm

Intuitive idea

- Directly apply fixed-point iteration to Bellman equation

How to find optimal value function $v^*(s)$

- View as an algebraic equation

$$v^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{a} \left(r_{ss'}^{a} + \gamma v^*(s') \right)$$

- Picard fixed-point iteration algorithm

Repeat k until to infinity

$$V_{k+1}(s) \leftarrow \max_{a} \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_k(s') \right), \forall s \in \mathcal{S}.$$

End

Convergence of Value Iteration Algorithm

- **Bellman operator is a γ-contraction mapping!**

$$\mathcal{B}\big(V(s)\big) \stackrel{\text{def}}{=} \max_a \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \left(r_{ss'}^a + \gamma V(s')\right)$$

- For an arbitrary state $s_{(i)} \in \mathcal{S}$

$$\left|\mathcal{B}\left(V_{k+1}(s_{(i)})\right) - \mathcal{B}\left(V_k(s_{(i)})\right)\right| = \left|\max_a \sum_{s' \in \mathcal{S}} \mathcal{P}\big(r + \gamma V_{k+1}(s')\big) - \max_a \sum_{s' \in \mathcal{S}} \mathcal{P}\big(r + \gamma V_k(s')\big)\right|$$

$$\leq \max_a \left|\sum_{s' \in \mathcal{S}} \mathcal{P}\big(r + \gamma V_{k+1}(s')\big) - \sum_{s' \in \mathcal{S}} \mathcal{P}\big(r + \gamma V_k(s')\big)\right|$$

Triangular inequality

$$= \gamma \max_a \left|\sum_{s' \in \mathcal{S}} \mathcal{P} V_{k+1}(s') - \sum_{s' \in \mathcal{S}} \mathcal{P} V_k(s')\right|$$

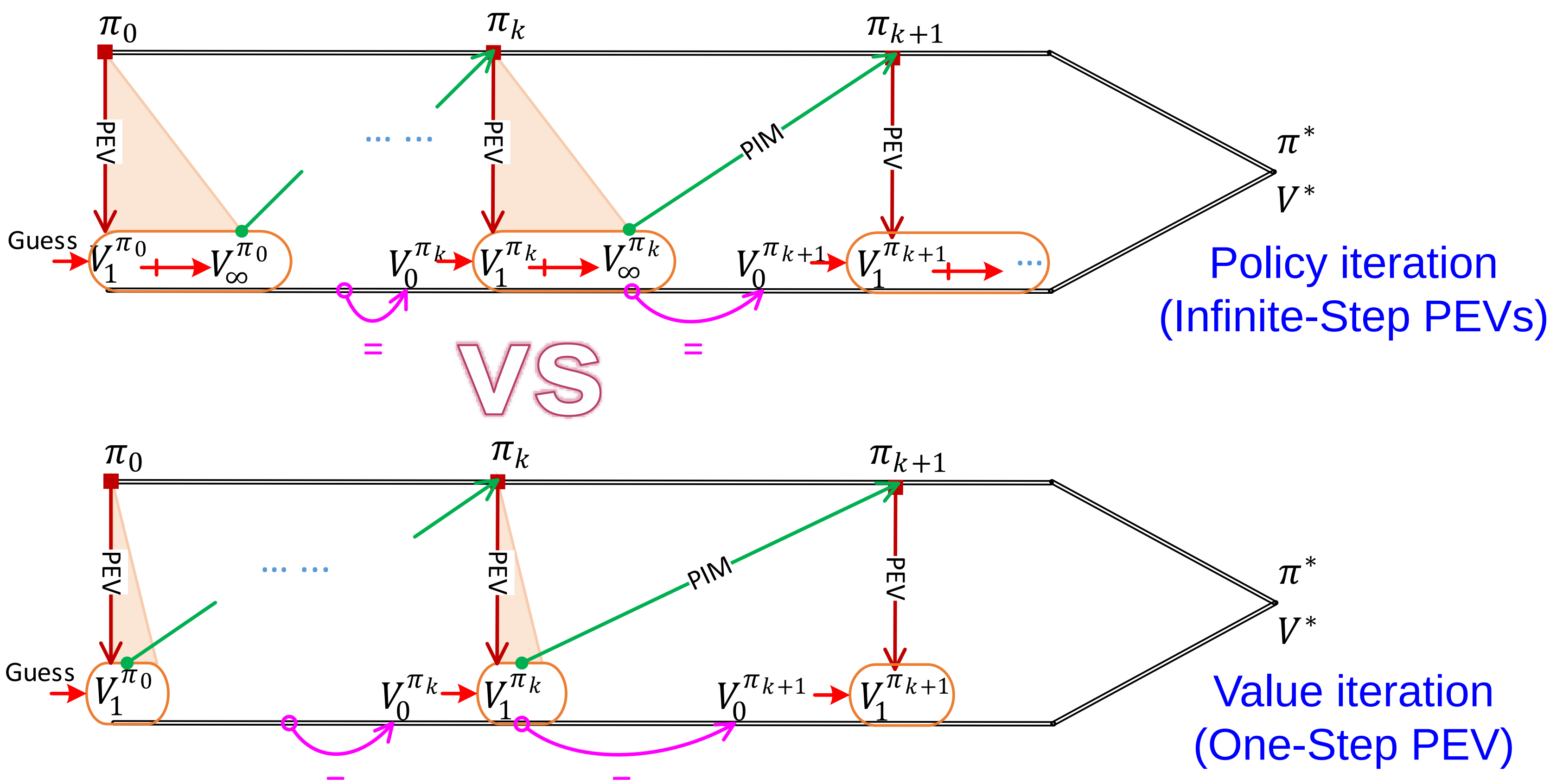
$$\leq \gamma \max_a \sum_{s' \in \mathcal{S}} \mathcal{P} \max_{s \in \mathcal{S}} \left|V_{k+1}(s) - V_k(s)\right|$$

$$= \gamma \|V_{k+1}(s) - V_k(s)\|_\infty$$

Take the ∞-norm of two sides for all the elements:

$$\left\|\mathcal{B}\big(V_{k+1}(s)\big) - \mathcal{B}\big(V_k(s)\big)\right\|_\infty \leq \gamma \|V_{k+1}(s) - V_k(s)\|_\infty$$

Unification of Policy Iteration & Value Iteration



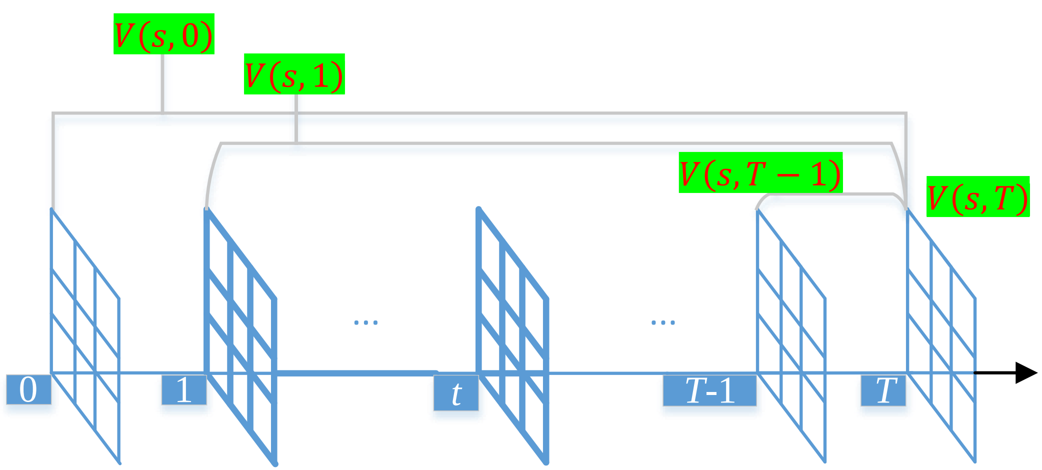
- Policy iteration & Value iteration are two extremes of generalized policy iteration (GPI), of which value iteration stops PEV after just one sweep, and policy iteration performs infinite numbers of PEVs

Value Iteration in Finite Horizon DP

Finite Horizon DP

- Optimal value function is dependent of time

$$V^*(s,t) = \max_{\pi}\left\{\sum_{i=0}^{T-t} r_{t+i} \,|s_t = s\right\}$$



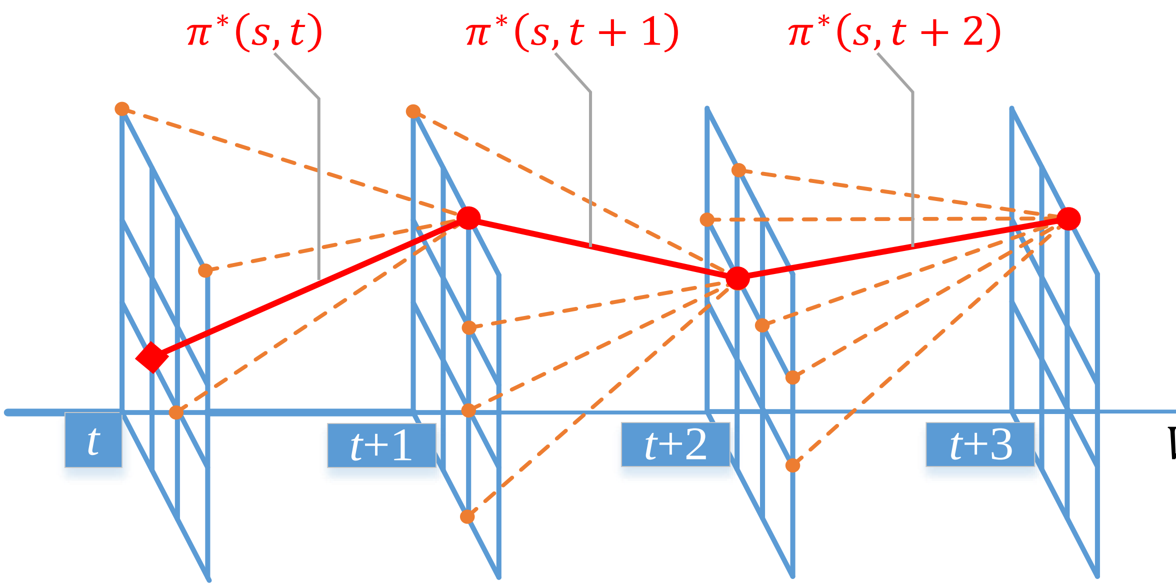
Bellman Equation

$$V^*(s,0) = \max\{r + V^*(s',1)\}$$
$$V^*(s,1) = \max\{r + V^*(s',2)\}$$
$$\cdots\ \cdots$$
$$V^*(s,t) = \max\{r + V^*(s',t+1)\}$$
$$\cdots\ \cdots$$
$$V^*(s,T-1) = \max\{r + V^*(s',T)\}$$
$$V^*(s,T) = \max\{r\}$$

Value Iteration in Finite Horizon DP

Solution: Exact DP

- Compute in a step-by-step backward manner



$$V^*(s,0) = \max\{r + V^*(s',1)\}$$

$$V^*(s,1) = \max\{r + V^*(s',2)\}$$

$$\dots\ \dots$$

$$V^*(s,t) = \max\{r + V^*(s',t+1)\}$$

$$\dots\ \dots$$

$$V^*(s,T-1) = \max\{r + V^*(s',T)\}$$

$$V^*(s,T) = \max\{r\}$$

- **Must be computed offline due to curse of dimensionality**

Connection with Finite Horizon DP

Exact DP in infinite horizon problem

- Value function of each stage becomes independent of time

$$V(s) \stackrel{\text{def}}{=} V^*(s,0) = V^*(s,1) = \cdots = V^*(s,t) = \cdots = V^*(s,\infty)$$

- Value functions of all stages are the same in structure
- Exact DP can degenerate into value iteration

$$\left.\begin{array}{c} V(s) \leftarrow \max\{r + V(s')\}, t = 0 \\ V(s) \leftarrow \max\{r + V(s')\}, t = 1 \\ \cdots \\ V(s) \leftarrow \max\{r + V(s')\}, t = \infty - 1 \\ V(s) = \max\{r\}, t = \infty \end{array}\right\rangle \Longrightarrow V(s) \leftarrow \max\{r + V(s')\}$$

Outline

1 Intro to Stochastic DP

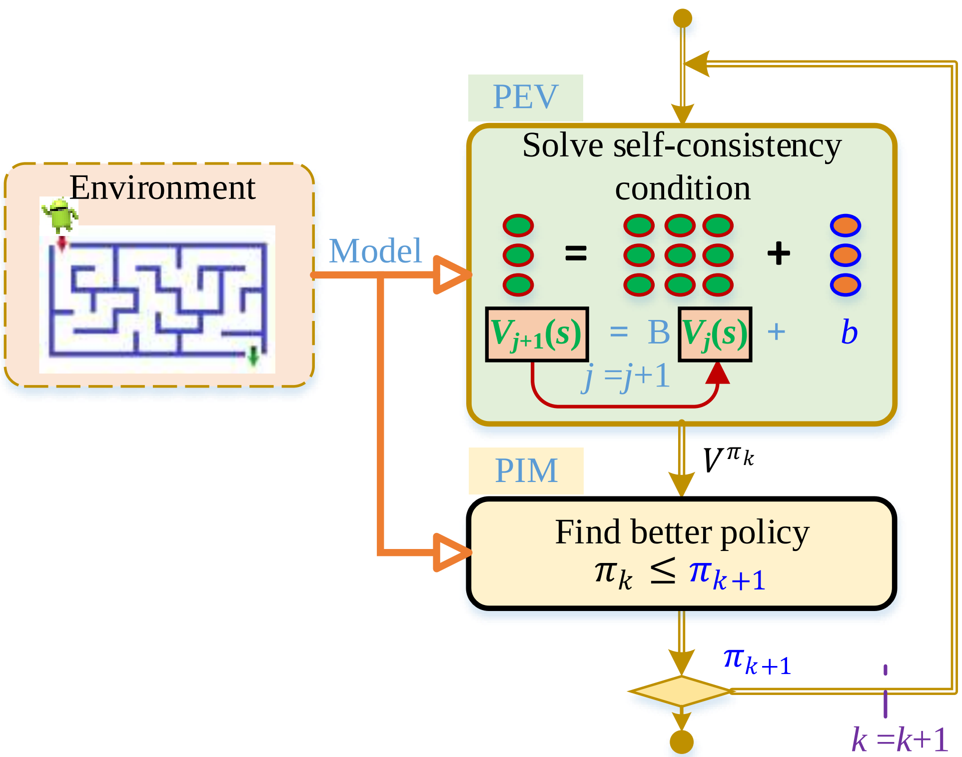
2 DP Policy Iteration

3 DP Value Iteration

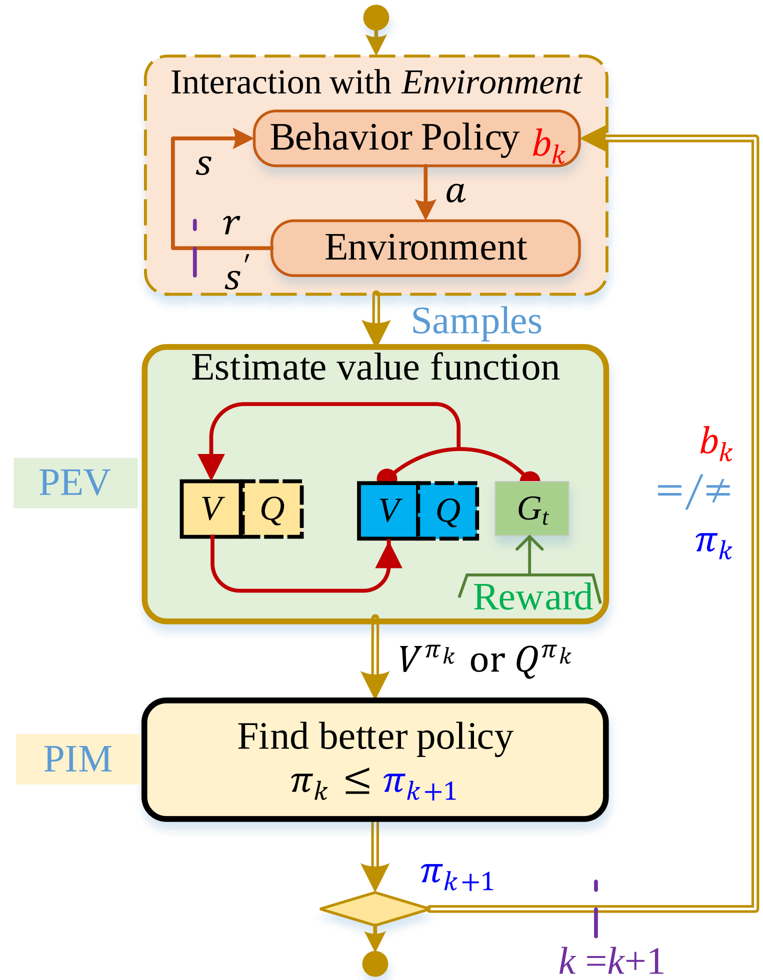
4 A Unified Framework

Unification of Model-free & Model-based

Model-based RL



Model-free RL



Unification of Model-free PEV & Model-based PEV

For model-free PEV:

- Using experience from interacting with environment
  - Average all the returns, like MC
  - Bootstrapping from existing estimate, like TD

For model-based PEV:

- Solve self-consistency condition with environment model
- Use fixed-point iteration method to find the root of self-consistency condition

Unification with fixed point explanation

Fixed-point iteration schemes

- (1) Picard iteration

$$X_n = f(X_{n-1})$$

- (2) Mann iteration

$$X_n = (1 - \alpha_n) X_{n-1} + \alpha_n f(X_{n-1})$$

- (3) Krasnoselskij iteration

$$X_n = (1 - \lambda) X_{n-1} + \lambda f(X_{n-1})$$

- (4) Ishikawa iteration

$$X_{n+1} = (1 - \beta_n) X_n + \beta_n f\big((1 - \alpha_n) X_{n-1} + \alpha_n f(X_{n-1})\big)$$

- (5) Kirk iteration

$$X_{n+1} = c_0 X_n + c_1 f(X_n) + c_2 f\big(f(X_n)\big) + \cdots + c_k f^k(X_n)$$

Unification with fixed point explanation

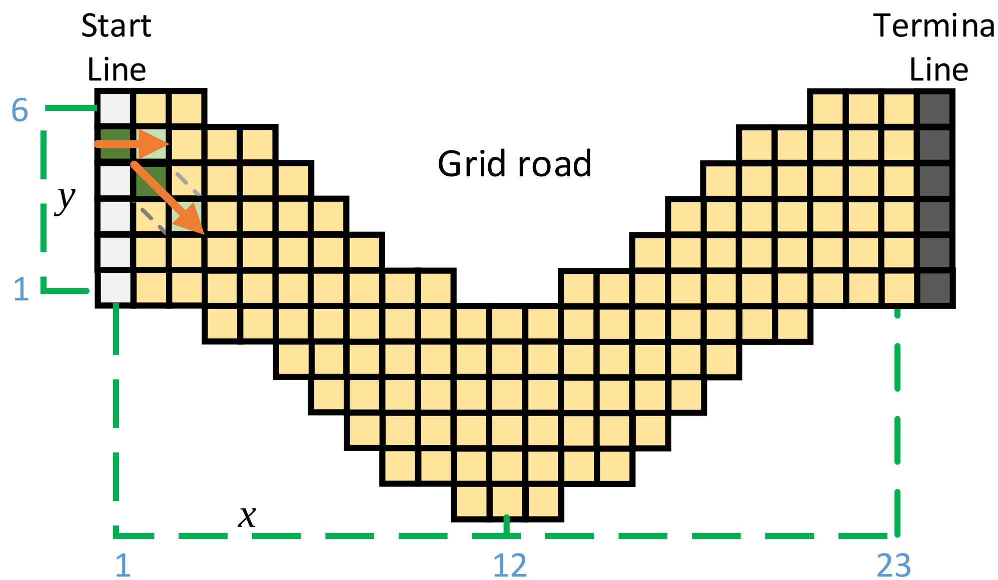
PEV in policy iteration

	Model-based	Model-free
Picard	PEV in DP policy iteration	--
Krasnoselskij	--	Sarsa, Expected Sarsa (with constant learning rate)
Mann	--	Sarsa, Expected Sarsa (with variable learning rate)
Ishikawa	--	--
Kirk	--	TD(n), TD-lambda

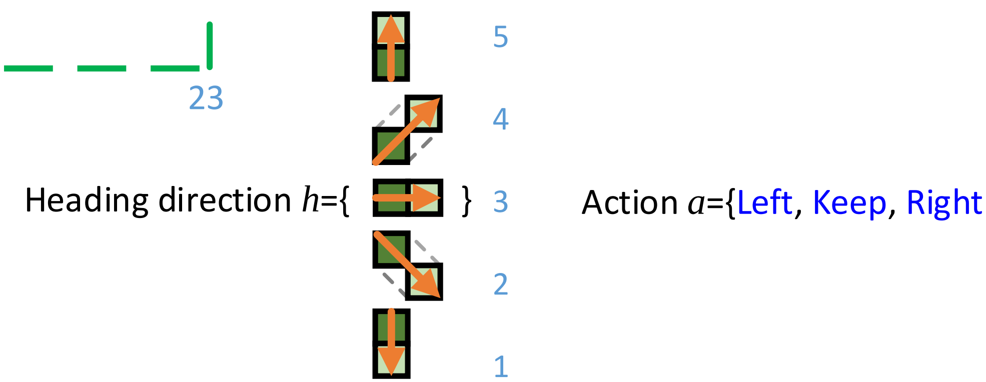
Value iteration

	Model-based	Model-free
Picard	DP value iteration	--
Krasnoselskij	--	Q-learning with constant learning rate
Mann	--	Q-learning with varying learning rate
Ishikawa	--	--
Kirk	--	--

Example: Autonomous Driving



Goal: An automated car passes the curved road quickly at the lowest energy consumption



- Each car is composed of two adjacent cells
- Every time instant, car will move one step forward along current heading direction

Example: Autonomous Driving

- **State space**

$$s = [x, y, h]^{\mathrm{T}} \in \mathcal{S}$$
$$\mathcal{S} = \mathcal{S}_x \times \mathcal{S}_y \times \mathcal{S}_h$$
$$\mathcal{S}_x = \{x_{(1)}, x_{(2)}, \cdots, x_{(23)}\}, \mathcal{S}_y = \{y_{(1)}, y_{(2)}, \cdots, y_{(6)}\}, \mathcal{S}_h = \{h_{(1)}, h_{(2)}, \cdots, h_{(5)}\}$$

- **Action space:**

$$\mathcal{A} = \{\mathrm{Left}, \mathrm{Keep}, \mathrm{Right}\}$$

  - Each action can deterministically steer the car to a neighboring direction, and move the car on-step forward along current direction

$$\Pr\left\{s' = \left[x_{(2)}, y_{(4)}, h_{(2)}\right]^{\mathrm{T}} \middle| \, s = \left[x_{(1)}, y_{(5)}, h_{(3)}\right]^{\mathrm{T}}, a = \mathrm{Right}\right\} = 1$$

Example: Autonomous Driving

Reward

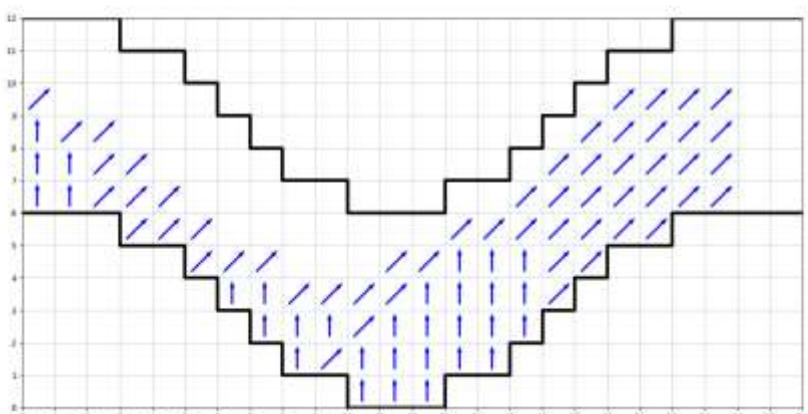
- Combine steering and moving rewards

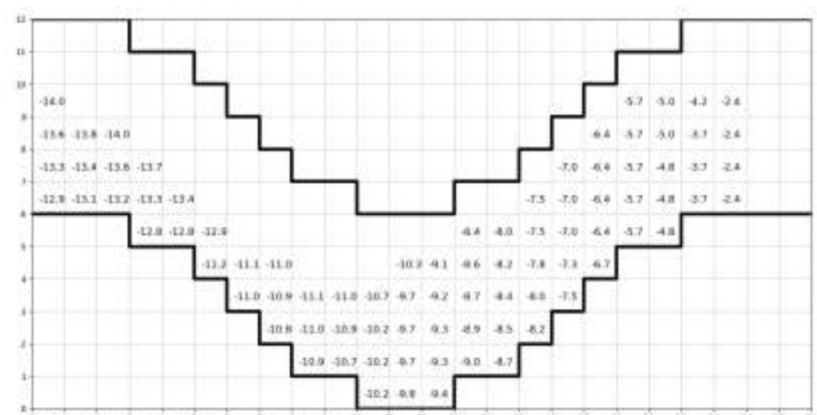
$$r(s,a,s') = r_{\text{Steer}} + r_{\text{Move}}$$

$$r_{\text{Steer}} = \begin{cases} -1 & \text{, if } a = \text{Left} \\ 0 & \text{, if } a = \text{Keep} \\ -1 & \text{, if } a = \text{Right} \end{cases} \qquad r_{\text{Move}} = \begin{cases} -1 & \text{, if } h' = 1 \text{ or } 3 \text{ or } 5 \\ -\sqrt{2} & \text{, if } h' = 2 \text{ or } 4 \end{cases}$$

Learned policy and value

- If all states have heading direction 5 in last step

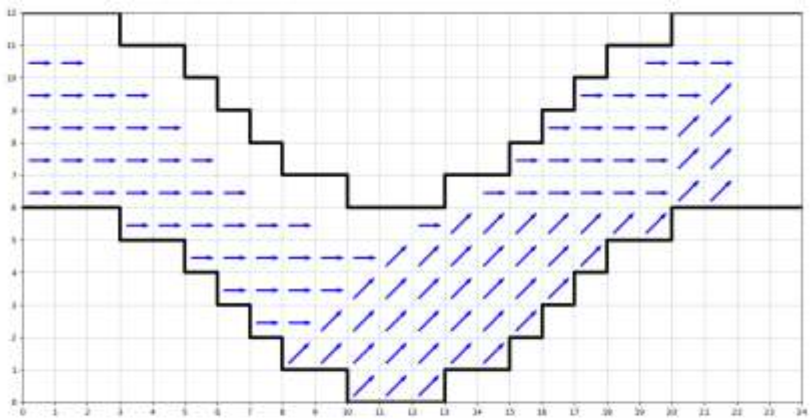


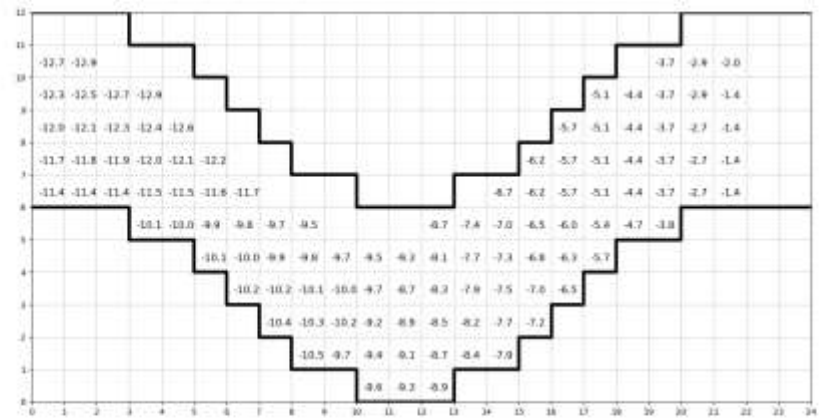


Example: Autonomous Driving

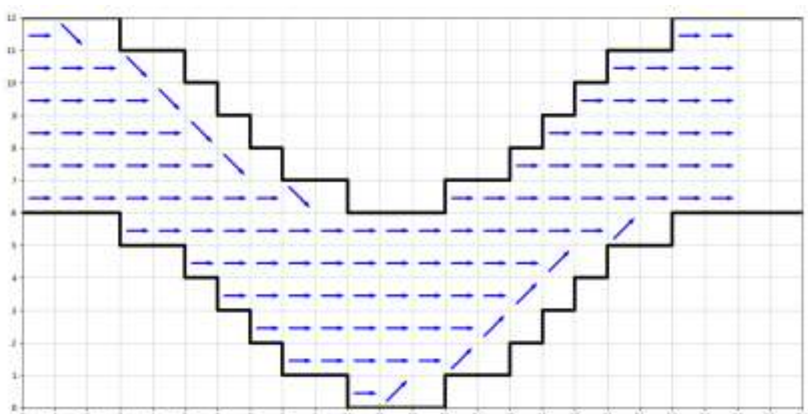
Learned policy and value

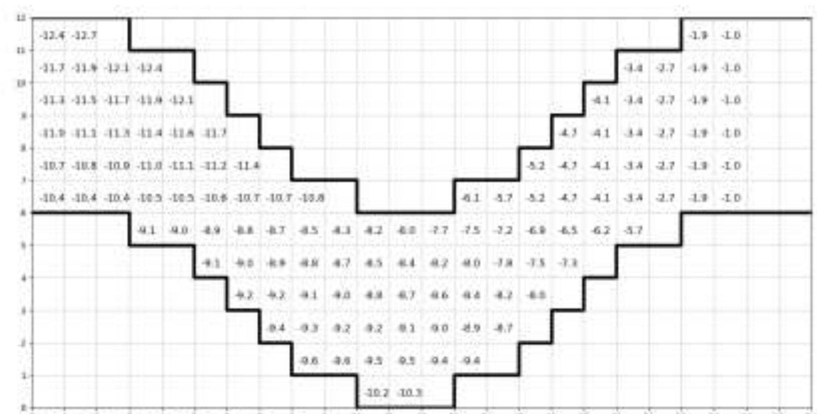
- If all states have heading direction 4 in last step





- If all states have heading direction 3 in last step

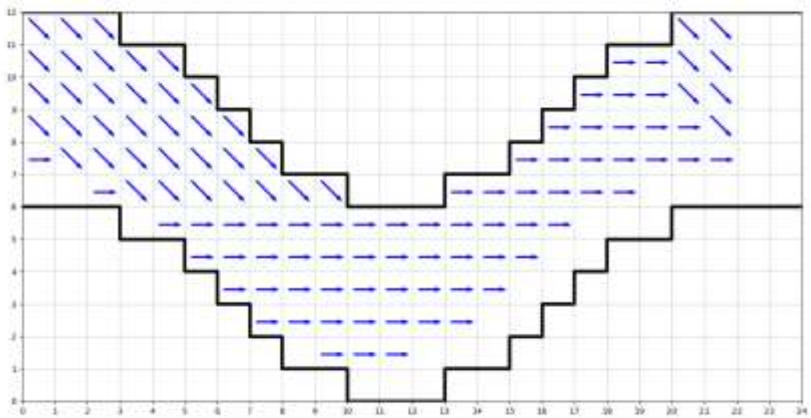
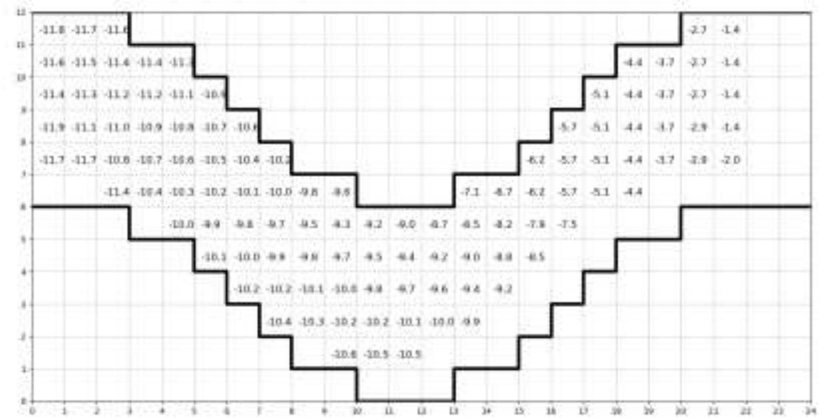




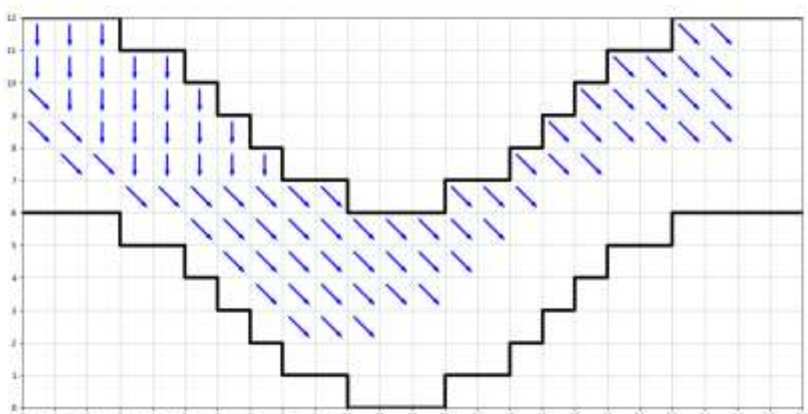
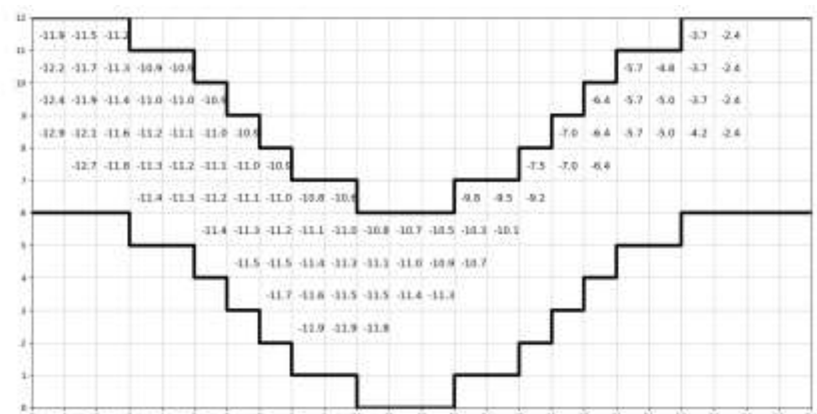
Example: Autonomous Driving

Learned policy and value

- If all states have heading direction 2 in last step

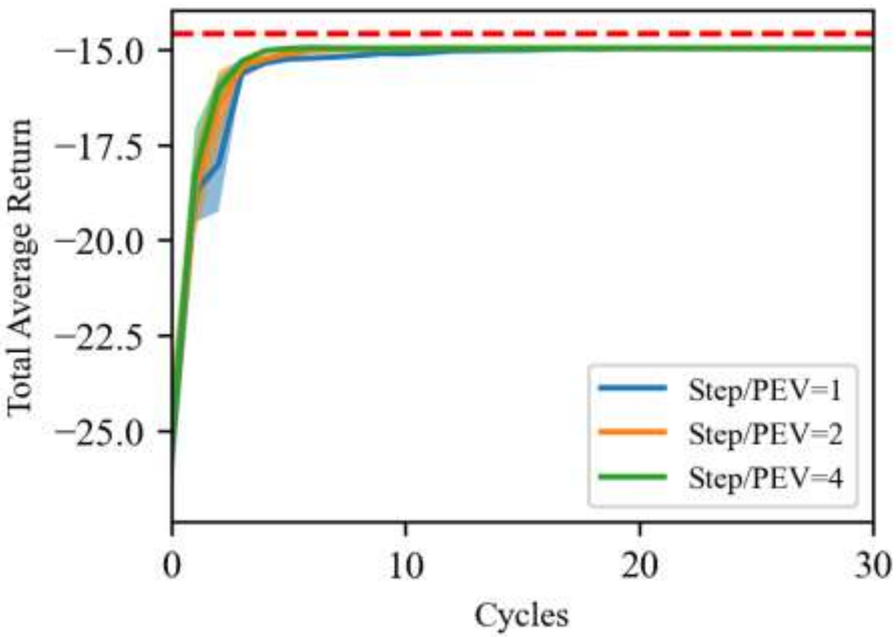



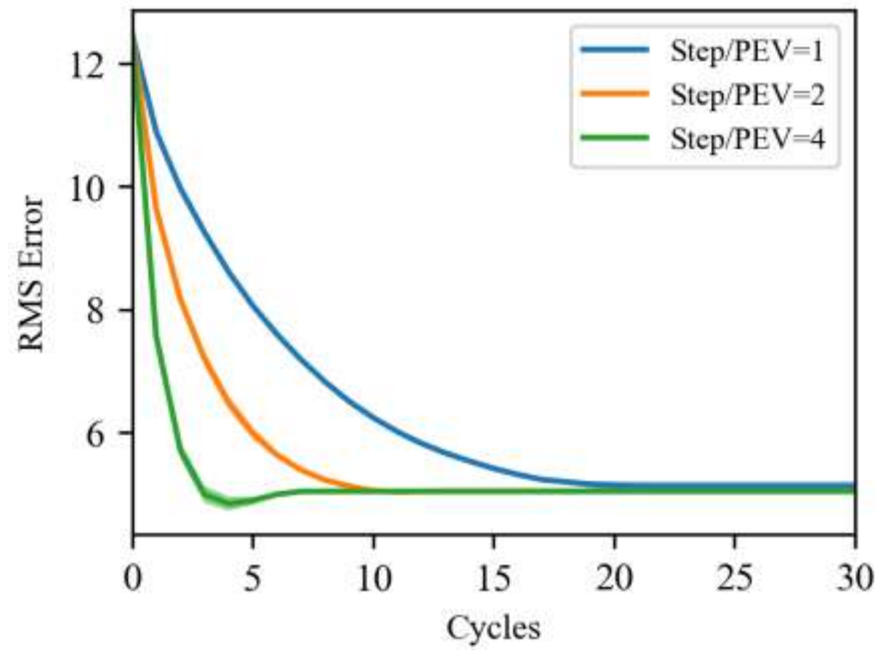
- If all states have heading direction 1 in last step

Example: Autonomous Driving

Influence of Step/PEV

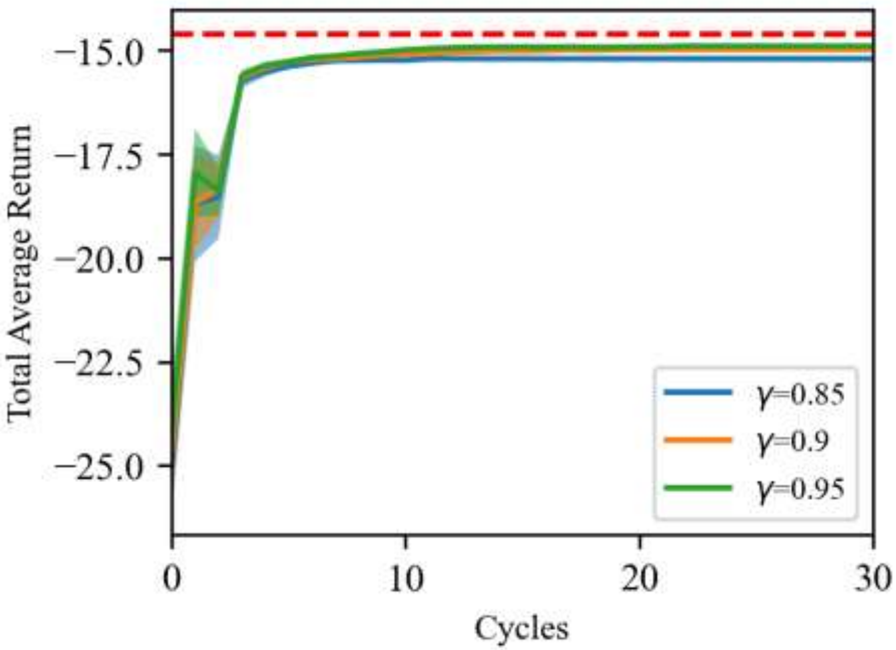


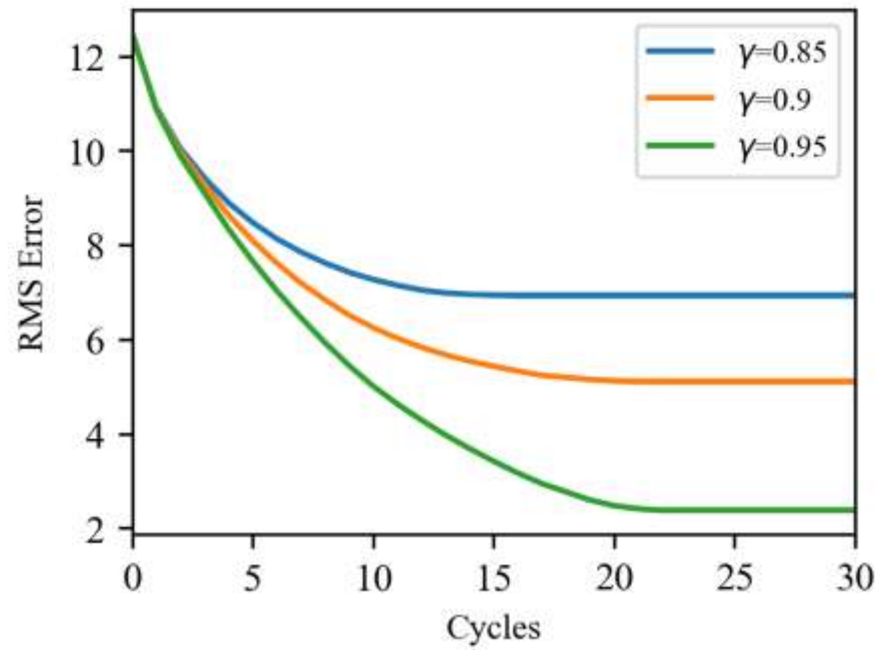


- With increasing PEV step size, RMS error converges more quickly
- Total average returns are almost the same at different Step/PEV

Example: Autonomous Driving

Influence of discount factor





- The higher the discount factor is, the more total average reward an RL agent receives
- A high discount factor pushes agent to consider long-term return

The End!