《强化学习与控制》

--

Temporal Difference RL

Shengbo Eben Li
(李升波)

Intelligent Driving Laboratory (iDLab)

Tsinghua University

Turtles all the way down!

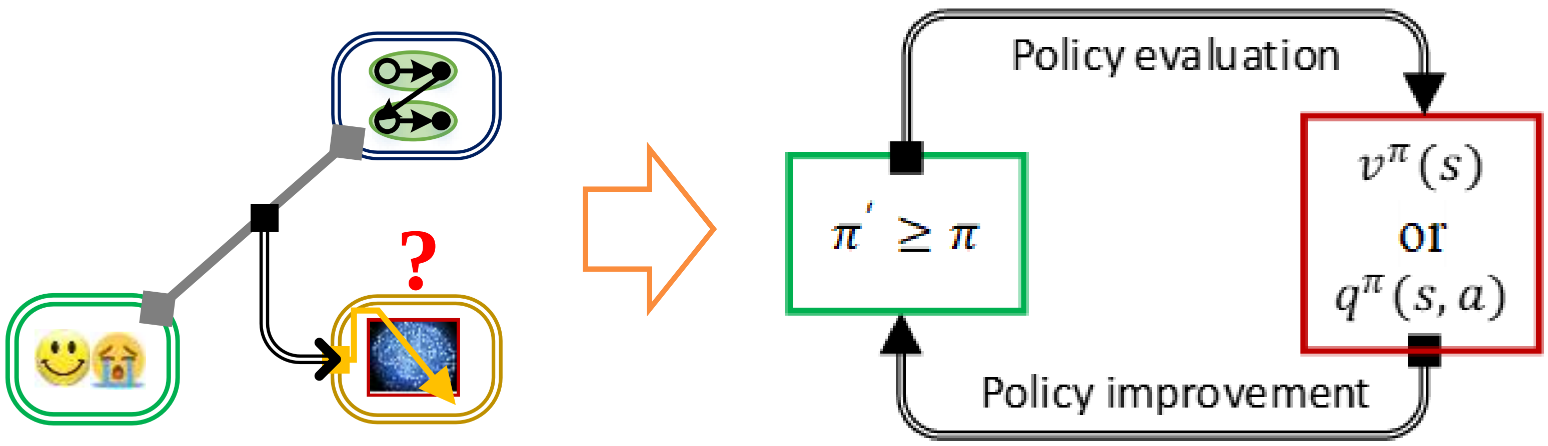
If I have seen further,
it is by standing on the shoulders of Giants.
-- Sir Isaac Newton (1643 - 1727)

It is turtles all the way down

Outline

1. TD Policy Evaluation
2. TD Policy Improvement
3. Typical TD Algorithms
4. Unified View of TD and MC

Alternating Cycle in Model-free RLs



A large class of RLs are to alternatively repeat two steps, i.e., PEV (Policy evaluation) and PIM (Policy improvement), so as to eventually find an optimal policy

Model-free RLs

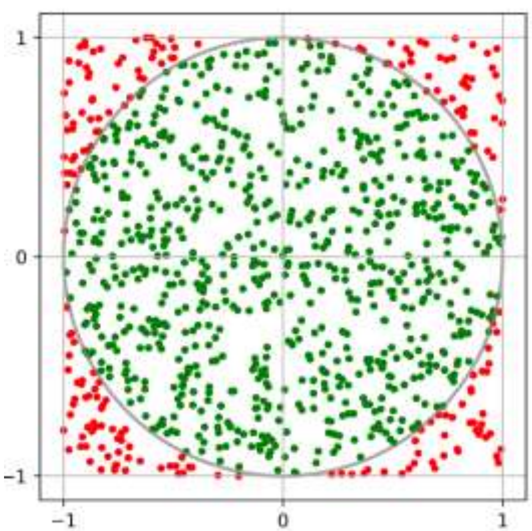
Key of Monte Carlo RL

- Method to estimate value function : estimate = average return

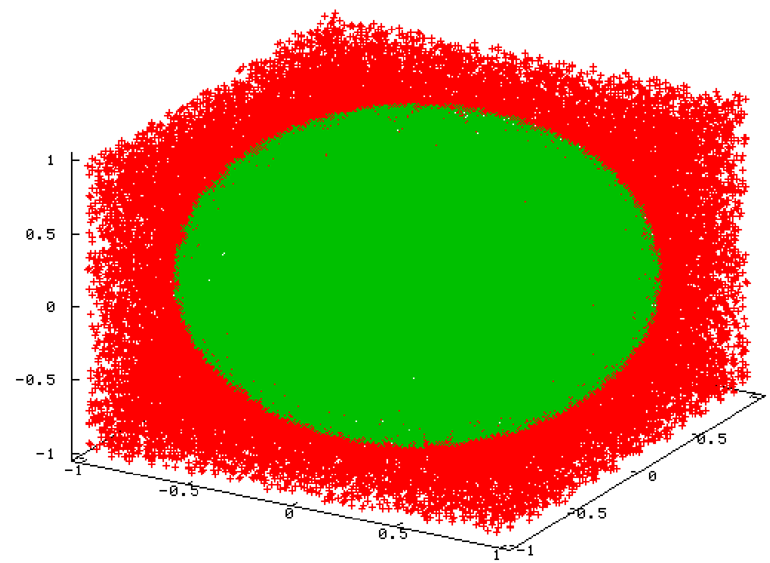
$$V^{\pi}(s) = \mathrm{Avg}\{G_{t:T} | s_t = s\}$$

$$Q^{\pi}(s,a) = \mathrm{Avg}\{G_{t:T} | s_t = s, a_t = a\}$$

Challenge



2D

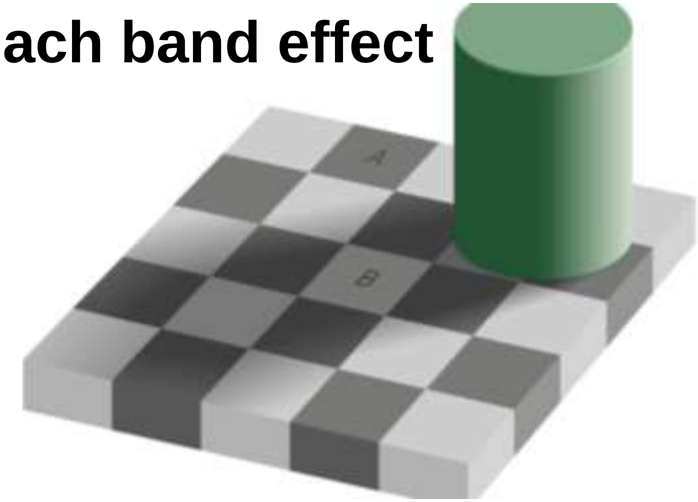


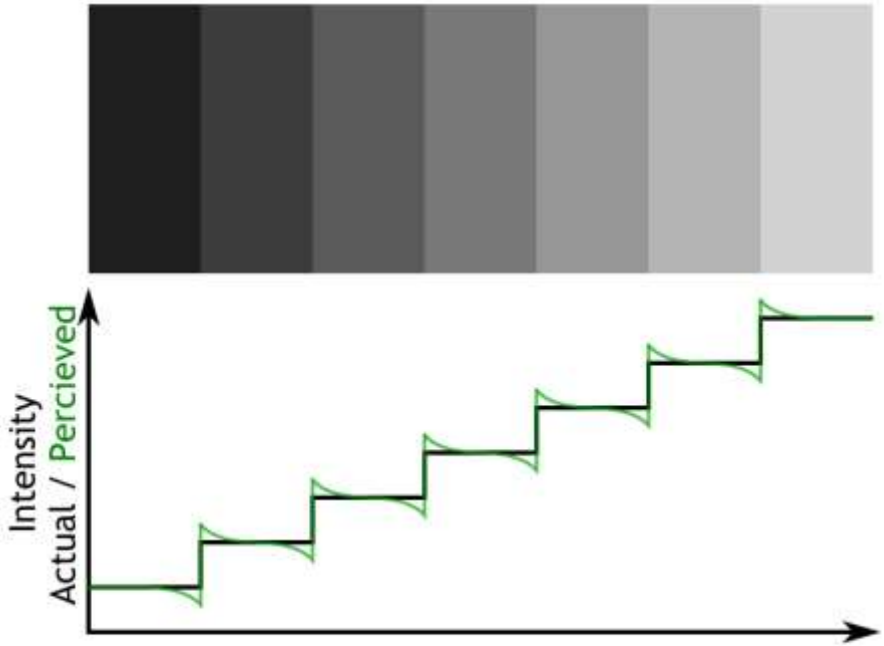
3D

Complexity increases exponentially with problem dimension

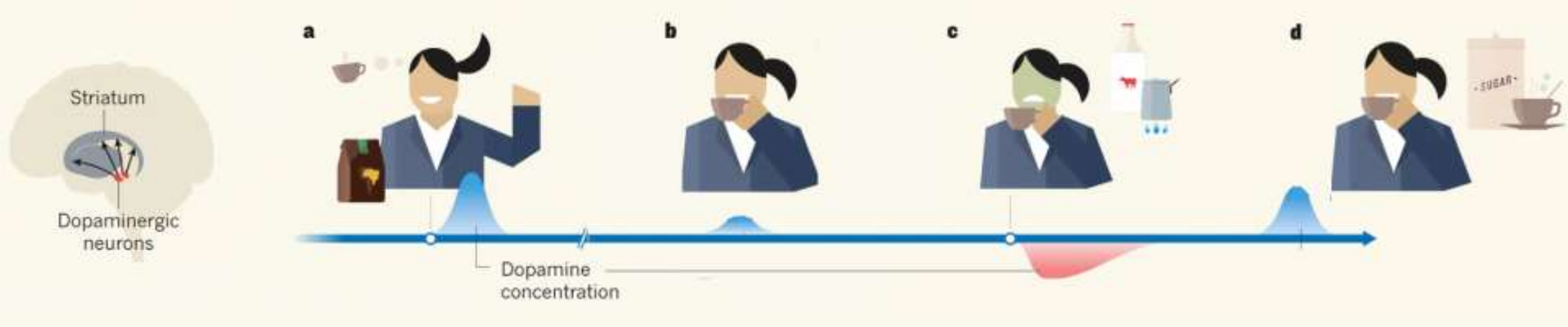
Reward Prediction Error Hypothesis

- **Mach band effect**





- **Reward Prediction Error vs Dopamine**



Temporal Difference RL

- **Temporal difference (TD) estimation**
  - **Bootstrapping**, i.e., learn by reusing existing estimates of value function
  - Suitable for both incomplete episodes and continuing tasks
  - Can learn in a step-by-step fashion, instead of episode-by-episode like MC

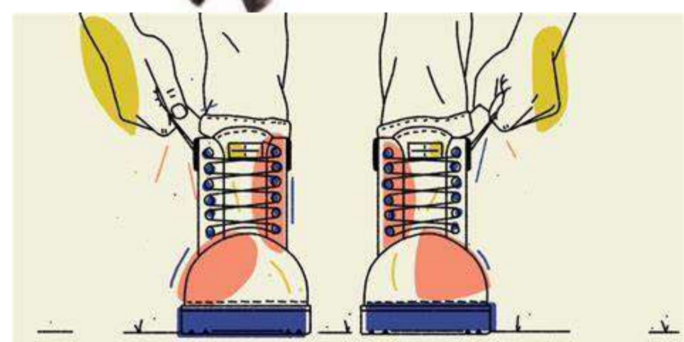
> If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference learning.
>
> ---- A. Barto & R. Sutton

Temporal Difference RL

“Bootstrapping” in Fiction

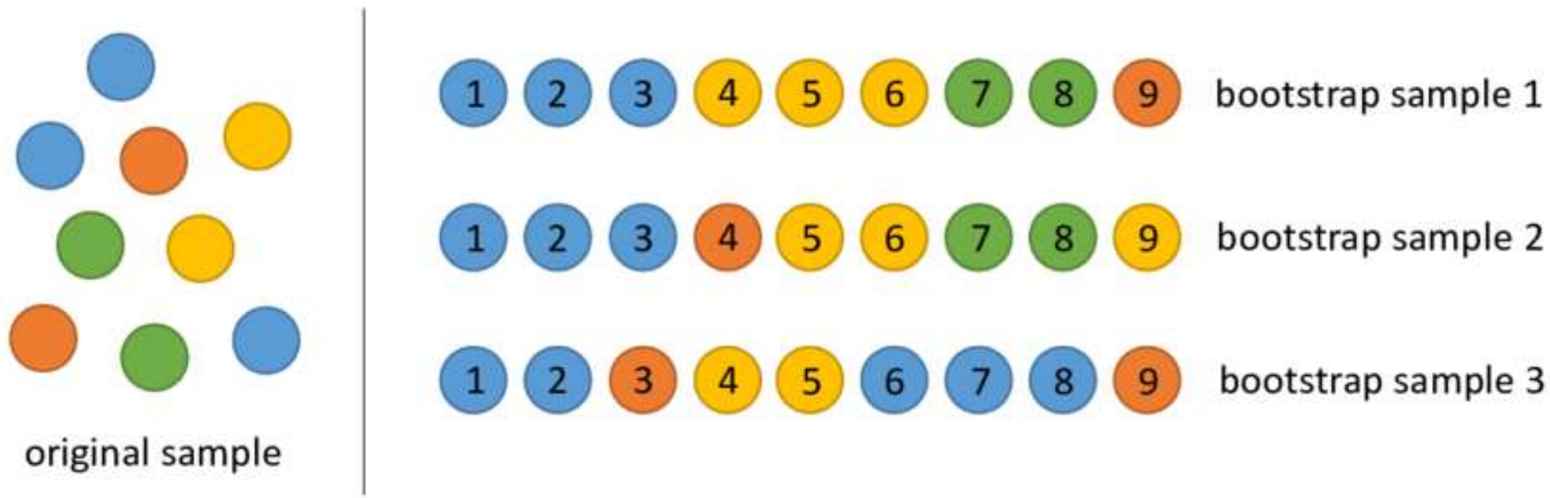
- “Baron pulls himself out of a swamp by his pigtail” -- By Rudolf Raspe in his story book <The Surprising Adventures of Baron Munchausen> (1781)

- [*idiom*] Pull oneself up by one’s own bootstraps



"Bootstrapping" in Science

- Developed in 1979 by Bradley Efron (an American statistician)
- A resampling method that estimates the sample statistics (mean, variance, percentiles) by drawing subsets randomly with replacement from a set of data points.



TD Policy Evaluation

Bootstrapping technique in TD

$$v^{\pi}(s_t) \leftarrow (1-\alpha)v^{\pi}(s_t) + \alpha \cdot G_t$$

Bootstrapping

Return : $G_t = \sum_{i=0}^{+\infty} \gamma^i r_{t+i}$

- Return G_t comes from interaction with environment

- Recall the self-consistency condition

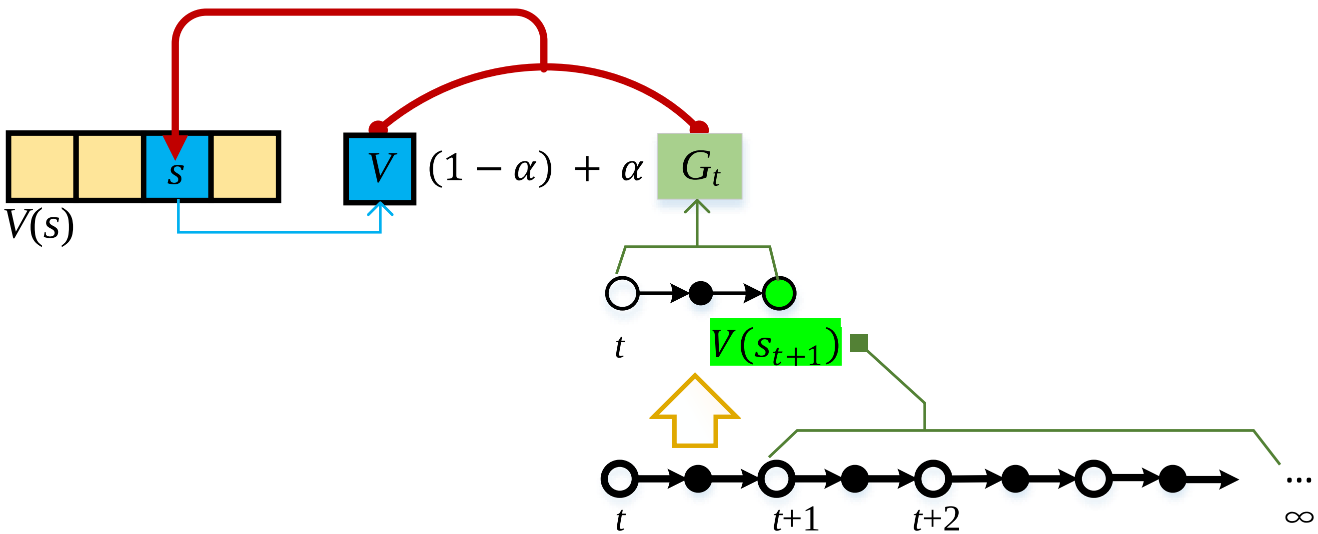
$$\mathbb{E}_{\pi}\{G_t|s\} = v^{\pi}(s) = \mathbb{E}_{\pi}\{r + \gamma v^{\pi}(s')\}$$

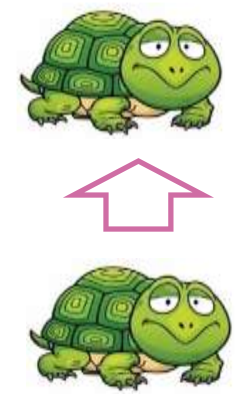
- Use estimate $V^{\pi}(s)$ to replace true value $v^{\pi}(s)$

$$\boxed{V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\big(\underbrace{r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)}_{\text{Reward Prediction Error}}\big)}$$

TD Policy Evaluation

One-step TD for State-value function





Use new sample (s_t, a_t, r_t, s_{t+1}) to update state-value $V(s)$

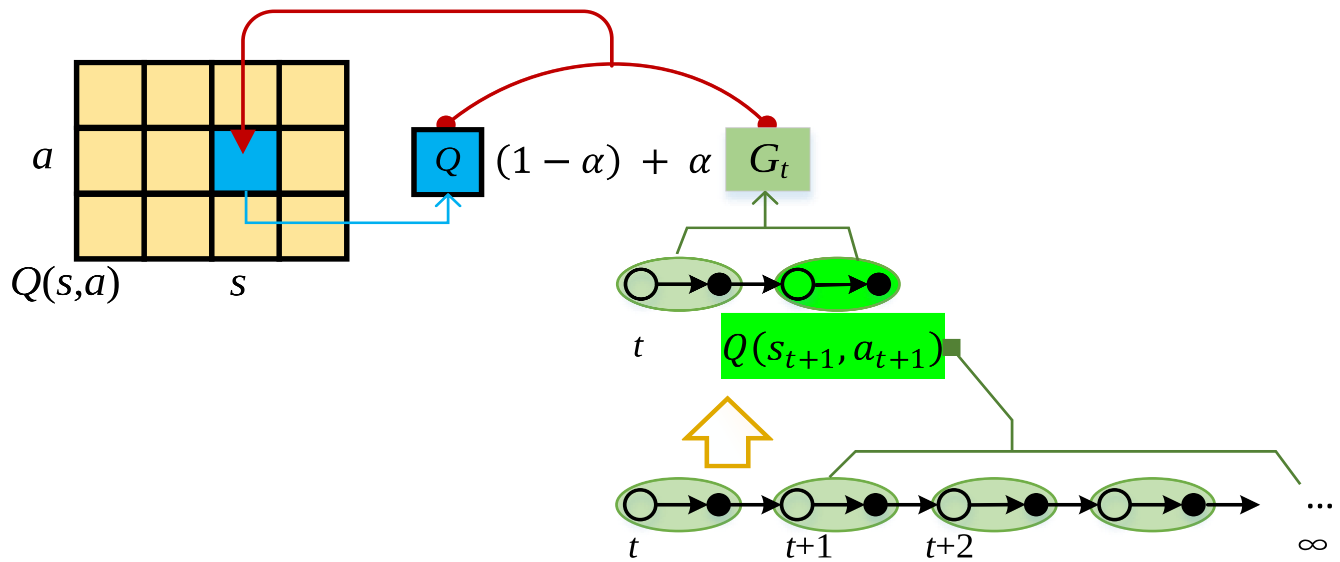
$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\left(r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)\right)$$

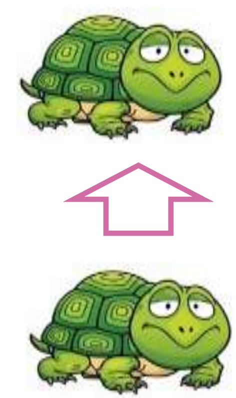
New Experience ← r_t

$V^\pi(s_t)$, $\gamma V^\pi(s_{t+1}) - V^\pi(s_t)$ → History estimate

TD Policy Evaluation

One-step TD for Action-value function





Use new sample $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ to update state-value $Q(s, a)$

$$Q^\pi(s_t, a_t) \leftarrow \underbrace{Q^\pi(s_t, a_t)}_{\text{History estimate}} + \alpha\big(\underbrace{r_t}_{\text{New Experience}} + \underbrace{\gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)}_{\text{History estimate}}\big)$$

New Experience

History estimate

TD Policy Evaluation

Advantages of TD Policy Evaluation

- Can update value function only by single sample (In comparison, MC must update at the end of each episode)

- Can be implemented in an online, fully incremental fashion

- Bootstrapping is helpful to increase estimate accuracy with only a few samples and accelerate convergence

Outline

TD Policy Improvement

On-policy strategy

- Use the same policy for sampler and learner
- Can use ϵ-greedy policy to increase exploitation
- Can shift to the optimal policy by gradually reducing ϵ

Off-policy strategy

- Use different policies in sampler and learner
  - Behavior policy b for more exploration in environment
  - Target policy π to determine the optimal policy
- Use importance sampling (IS) technique to estimate value function under π by using samples from b

TD Policy Improvement

Importance sampling in Off-policy TD

- Define one-step IS ratio as

$$\rho_{t:t} = \frac{d_\pi(a_t, s_{t+1})}{d_b(a_t, s_{t+1})} = \frac{\pi(a_t|s_t)}{b(a_t|s_t)}$$

- Expected return under π can be estimated by using samples generated by b

$$\begin{aligned}
v^{\boldsymbol{\pi}}(s) &= \mathbb{E}_\pi\{r_t + \gamma v^\pi(s_{t+1})\} \\
&= \sum_{(a_t, s_{t+1}) \in \mathcal{A} \times \mathcal{S}} \frac{d_\pi(a_t, s_{t+1})}{d_b(a_t, s_{t+1})} d_b(a_t, s_{t+1}) \big(r_t + \gamma v^\pi(s_{t+1})\big) \\
&= \sum_{(a_t, s_{t+1}) \in \mathcal{A} \times \mathcal{S}} d_b(a_t, s_{t+1}) \Big(\rho_{t:t}\big(r_t + \gamma v^\pi(s_{t+1})\big)\Big) \\
&= \mathbb{E}_b\big\{\rho_{t:t}\big(r_t + \gamma v^\pi(s_{t+1})\big)\big\}
\end{aligned}$$

TD Policy Improvement

Importance sampling in Off-policy TD

- One-step TD prediction with IS ratio

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\left(\boldsymbol{\rho_{t:t}}\left(\boldsymbol{r_t} + \boldsymbol{\gamma V^{\pi}(s_{t+1})}\right) - V^{\pi}(s_t)\right)$$

- Often has low quality because of its high variance

- A short explanation: $\boldsymbol{\rho_{t:t}} \rightarrow \infty$ if $b(a|s) \rightarrow 0$
  - Sample becomes very important, and its noise is infinitely amplified
  - Bootstrapping mechanism loses effect

TD Policy Improvement

Importance sampling in Off-policy TD

- Method for variance reduction

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\left(\rho_{t:t}\left(r_t + \gamma V^\pi(s_{t+1})\right) - V^\pi(s_t)\right)$$

$$\Downarrow$$

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\left(\rho_{t:t}\underbrace{\left(r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)\right)}_{\text{One-step TD error}}\right)$$

$\rho_{t:t}$ $\times$ One-step TD error

The amplification effect from uncertain IS ratio is decreased

- Key of Proof

$$\mathbb{E}_b\{V^\pi(s)\} = \mathbb{E}_b\{\rho_{t:t}V^\pi(s)\}$$

Importance sampling in Off-policy TD

$$\mathbb{E}_b\{V^\pi(s)\} = \mathbb{E}_b\{\rho_{t:t} V^\pi(s)\}$$

- Proof:
  - $\mathbb{E}_b\{\cdot\}$ is an abbreviation of $\mathbb{E}_b\{\cdot\,|s\}$ for brevity
  - $V^\pi(s)$ can be viewed as a constant because s is known

$$\mathbb{E}_b\{\rho_{t:t} V^\pi(s)\} = \mathbb{E}_{a\sim b}\left\{\frac{\pi(a|s)}{b(a|s)}\right\} \mathbb{E}_{a\sim b}\{V^\pi(s)\}$$

$$= \sum_a b(a|s) \left.\frac{\pi(a|s)}{b(a|s)}\right|_s \cdot \mathbb{E}_{a\sim b}\{V^\pi(s)\}$$

$$= \left.\sum_a \pi(a|s)\right|_s \mathbb{E}_{a\sim b}\{V^\pi(s)\}$$

$$= 1 \cdot \mathbb{E}_b\{V^\pi(s)\}$$

Outline

1. TD Policy Evaluation
2. TD Policy Improvement
3. Typical TD Algorithms
4. Unified View of TD and MC

Common TD algorithms

- **(1) Sarsa**
  - Rummery & Niranjan, 1994
- **(2) Expected Sarsa**
  - Sutton, 1994
- **(3) Q-learning**
  - Watkins, 1989 (algorithm)
  - Watkins & Dayan, 1992 (convergence)

Mahesan Niranjan
(University of Southampton)

Chris Watkins
(University of London)

Sarsa

Sarsa (State-action-reward-state-action)

- On-policy one-step TD
- Update whenever encounter a 5-tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$

Policy Evaluation (PEV)

- Estimate action-value function $Q(s,a)$ with N samples

Loop N times

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha\big(r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)\big)$$

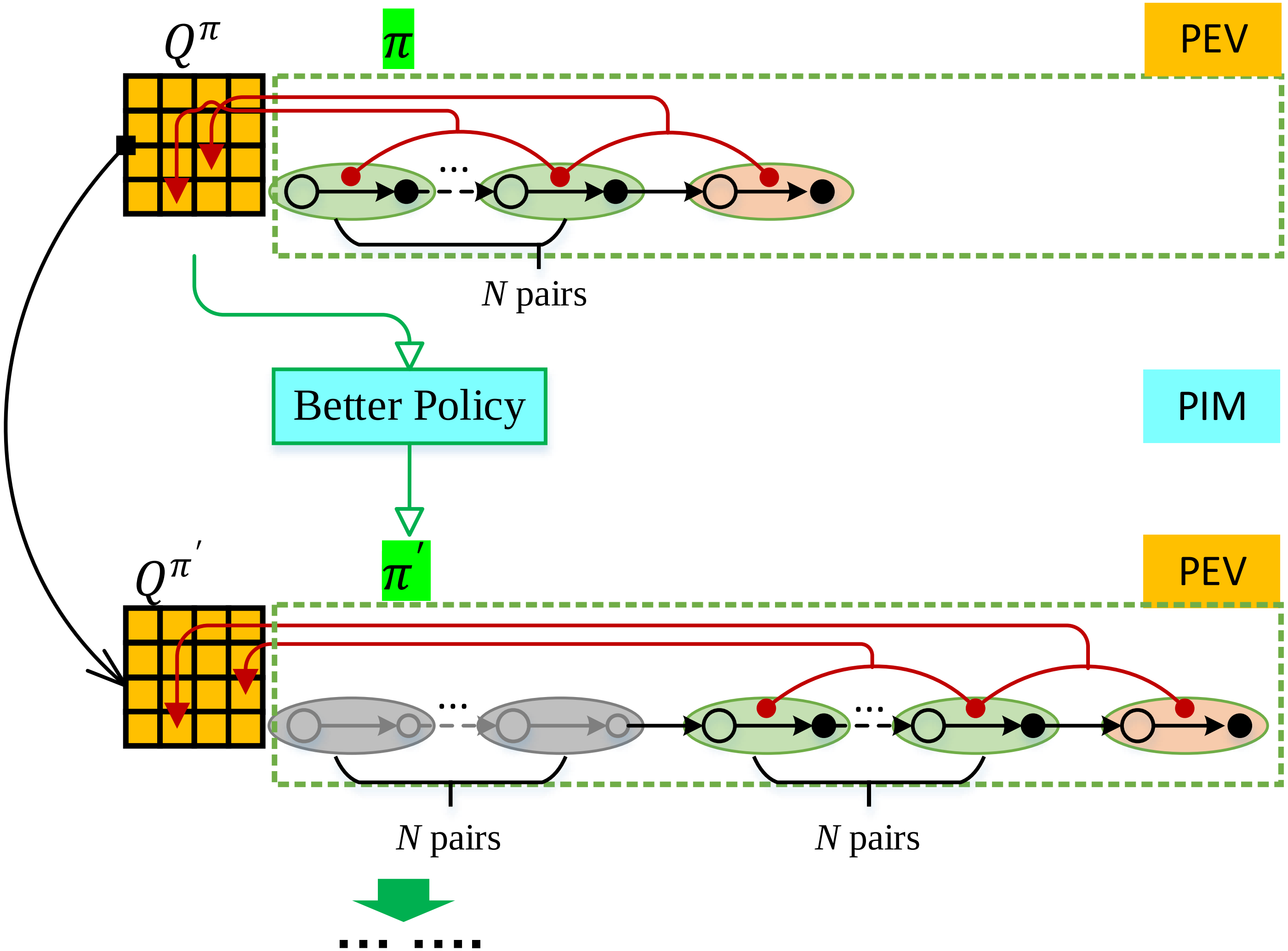
End

Policy Improvement (PIM)

- Find a better policy satisfying policy improvement theorem
- Update **ϵ-greedy policy** with respect to action-value function

Sarsa

Flow chat of Sarsa



Pseudocode of Sarsa

Hyperparameters: Discount factor γ, exploration rate ϵ, learning rate α, pairs per PEV N

Initialization: $Q(s,a) \leftarrow 0$, $\pi^{\epsilon}(a|s) \leftarrow 1/|\mathcal{A}|$

Repeat (indexed with k)

    // Environment initialization

    $s_0 \sim d_{\text{init}}(s)$

    $a_0 \sim \pi^{\epsilon}(a|s_0)$

    $s \leftarrow s_0, a \leftarrow a_0$

    Repeat until each episode terminates

        Rollout one step with action a, observe next state s' and reward r

        Sample action a' under next state s': $a' \sim \pi^{\epsilon}(a|s')$

        //Do action-value update

        $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$

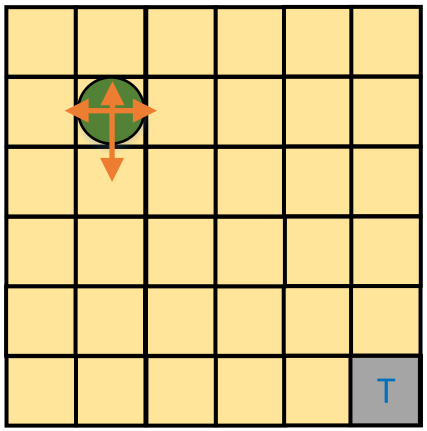
        If N steps since last policy update, update $\pi^{\epsilon}(a|s)$ with $Q(s,a)$

        $s \leftarrow s', a \leftarrow a'$

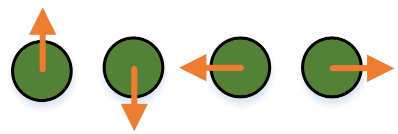
    End

End

Example: Clean Robot

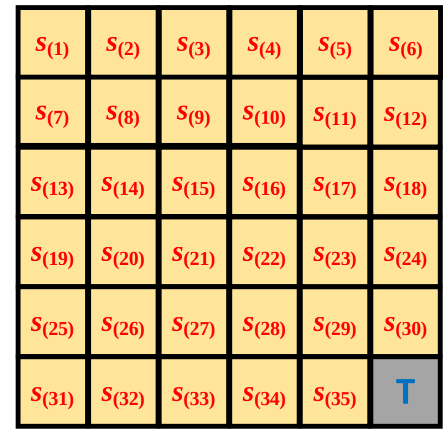


Grid environment



Action = {Up, Down, Left, Right}

Action space



State = $\{s_{(i)}\}$, i=1, 2,… , 36

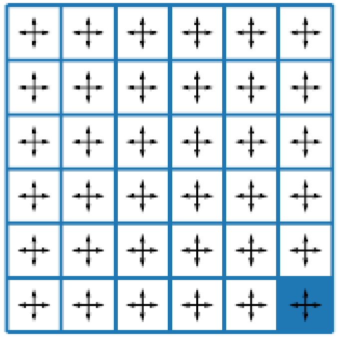
State space

$$\Pr\{s' = \text{Front Cell} \mid s, a\} = 0.8$$
$$\Pr\{s' = \text{Left Cell} \mid s, a\} = 0.1$$
$$\Pr\{s' = \text{Right Cell} \mid s, a\} = 0.1$$
$$\Pr\{s' = \text{Back Cell} \mid s, a\} = 0$$

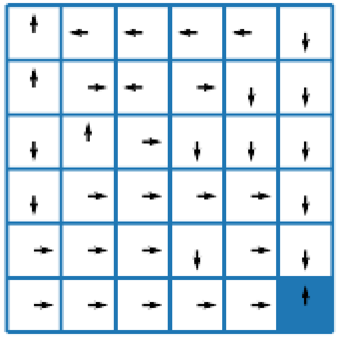
$$r(s, a, s') = \begin{cases} -1 & \text{if } s' \neq \mathrm{T} \\ +9 & \text{if } s' = \mathrm{T} \end{cases}$$

Example: Clean Robot

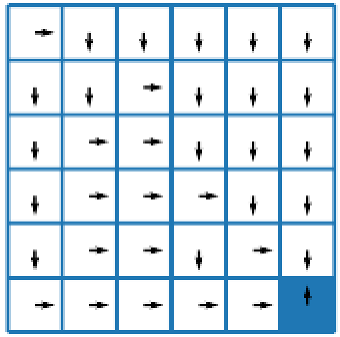
Improved Policy



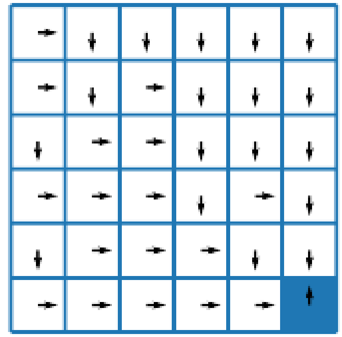
k=0



k=200

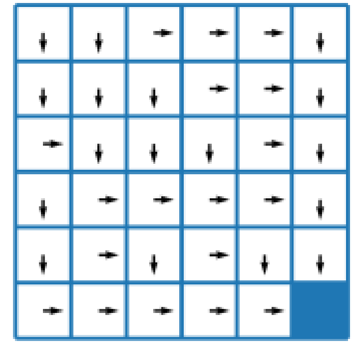


k=500



k=2000

Final Policy



Estimated state value

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	0.0	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	0.0	0.0	

k=0

-2.9	-2.8	-2.7	-2.3	-2.1	-1.4
-3.0	-2.8	-2.6	-2.1	-1.1	1.3
-2.8	-2.6	-2.1	-0.6	1.2	3.9
-2.4	-2.0	-0.5	2.0	4.0	6.6
-2.0	-1.3	0.8	3.8	5.8	8.5
-1.8	-0.9	2.0	5.1	8.2	

k=200

-3.9	-3.7	-3.4	-2.7	-1.6	-0.3
-3.7	-3.2	-2.3	-0.7	0.3	2.3
-3.2	-2.2	-1.1	0.5	1.8	4.3
-2.8	-0.9	0.8	2.2	3.5	6.0
-2.0	0.3	1.5	3.6	6.8	8.7
-0.7	1.6	3.9	5.2	7.6	

k=500

-4.5	-3.7	-2.8	-1.4	-0.7	0.4
-3.8	-2.5	-1.5	-0.2	1.5	2.9
-2.8	-1.4	-0.2	1.1	2.0	4.4
-1.7	-0.7	1.2	2.7	4.6	6.0
-1.3	0.7	2.5	4.4	6.5	8.2
-0.7	1.4	3.4	5.8	8.7	

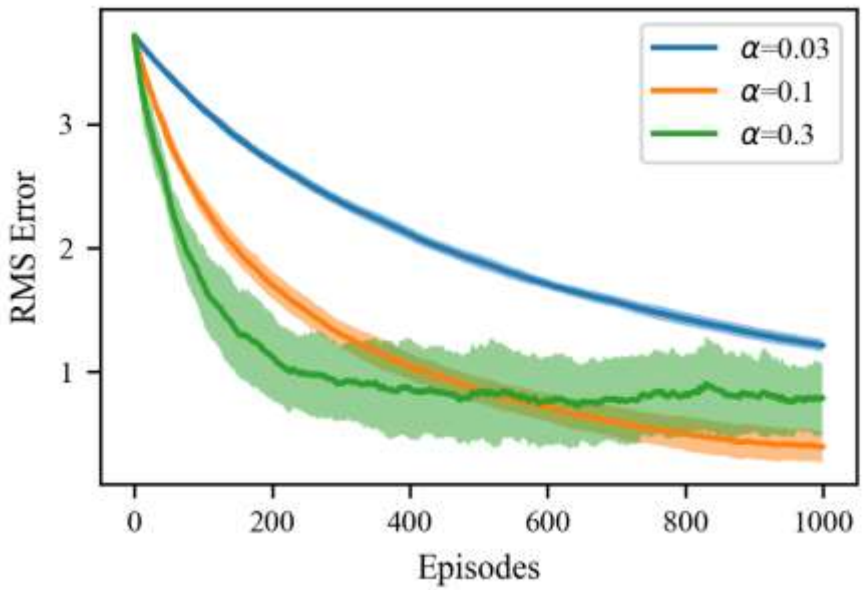
k=2000

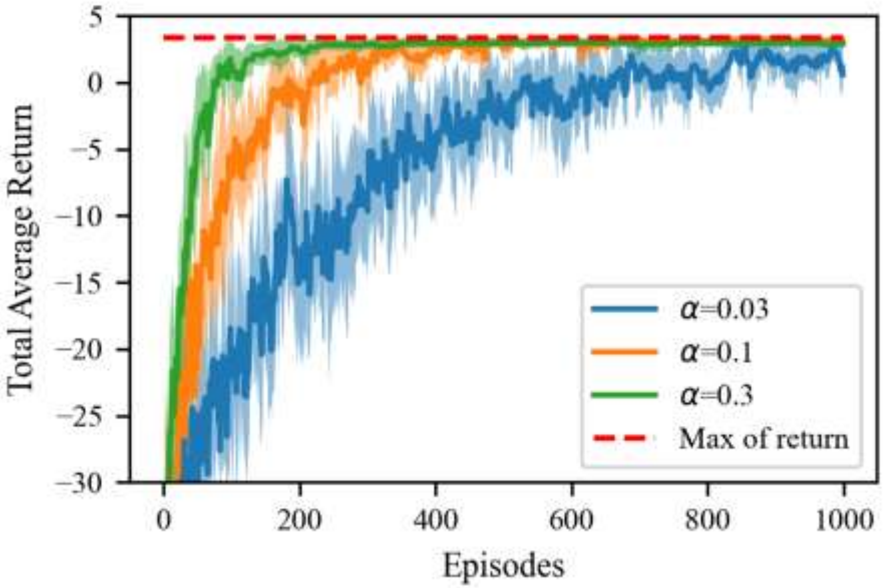
Final Value

-4.9	-4.3	-3.1	-1.8	-0.5	1.3
-4.1	-3.1	-2.0	-0.5	1.1	2.4
-3.1	-2.2	-1.1	0.6	2.6	3.9
-1.8	-0.5	0.4	2.6	4.0	5.8
-0.9	0.5	2.1	4.6	6.7	8.0
0.0	1.2	4.3	6.8	8.9	

Performance of Sarsa

- Influence of different learning rate



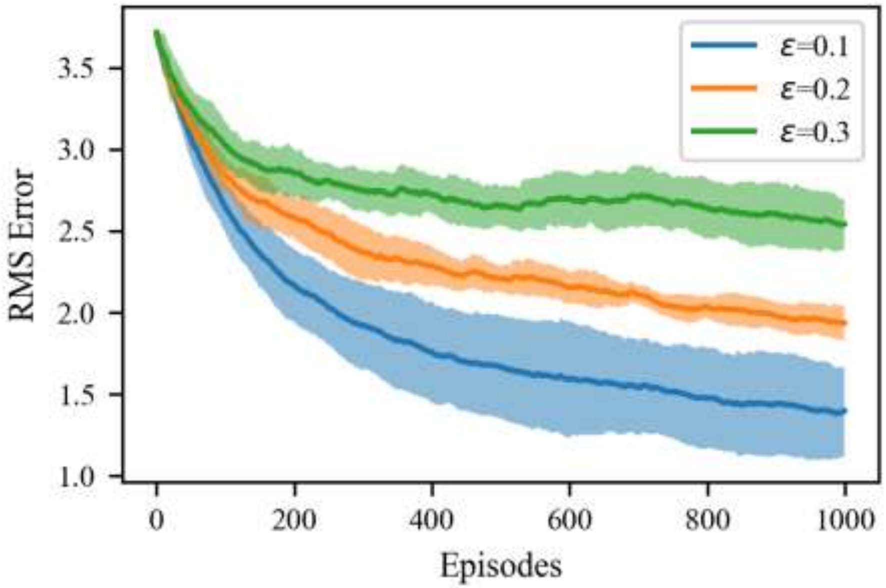


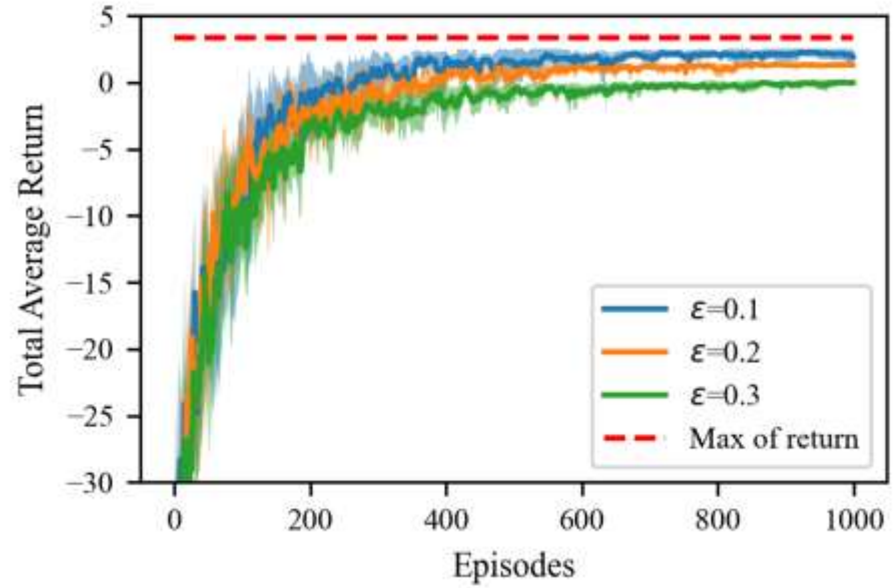
Learning rate can affect convergence speed largely, but eventually reach the same total average return

Sarsa

☐ Performance of Sarsa

- Influence of different exploration rate

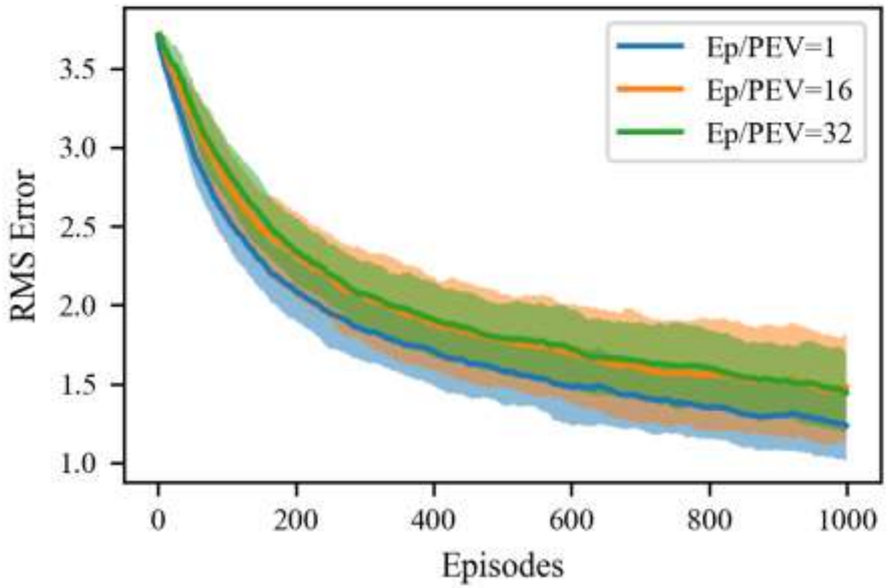


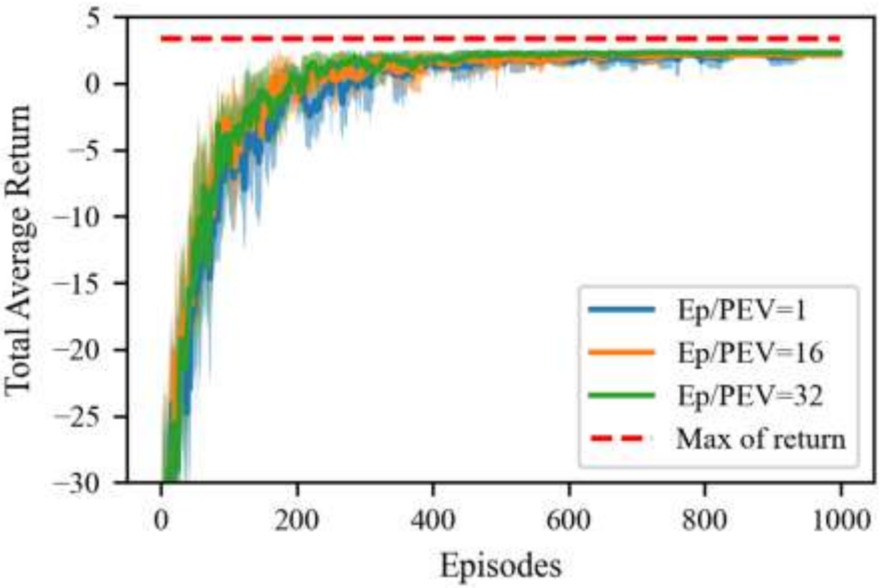


An appropriate exploration rate is essential for trade-off of exploration and exploitation

Performance of Sarsa

- Influence of number of used pairs in each PEV





The size of pairs/PEV has no obvious influence on policy performance

Q-learning

- One of the most important breakthroughs in TD
- A special variant of off-policy TD
  - Simple in form and easy to prove convergence
- Equivalent to one-step off-policy TD without explicit IS ratio

Play Super-Mario in Atari 2600 game with Deep Q-Network

Q-learning

- Only one update formula

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right)$$

- PEV: one step TD estimation

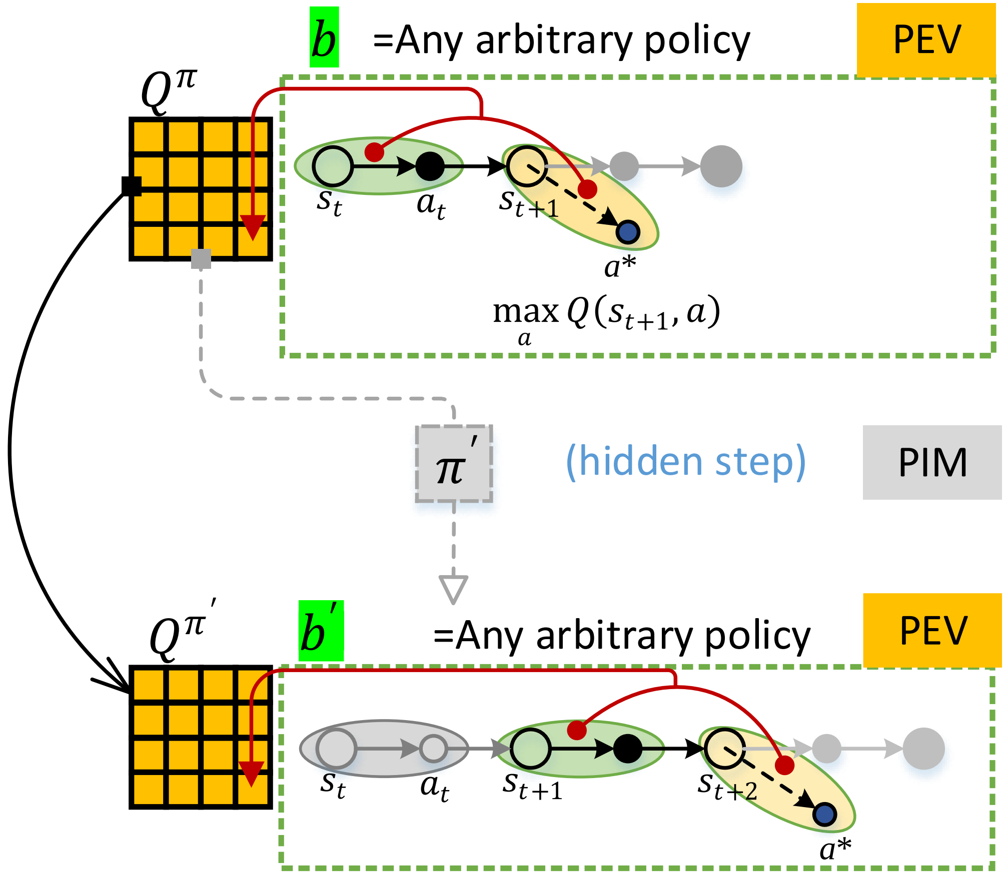
$$Q^\pi(s_t,a_t) \leftarrow Q^\pi(s_t,a_t) + \alpha\left(r_t + \gamma Q^\pi\left(s_{t+1}, \boldsymbol{\pi}(\boldsymbol{s}_{t+1})\right) - Q^\pi(s_t,a_t)\right)$$

$$\boldsymbol{a}_{t+1} = \boldsymbol{\pi}(\boldsymbol{s}_{t+1})$$

- PIM: next policy = greedy policy

$$\pi'(s) = \arg\max_a Q^\pi(s,a), \forall s \in \mathcal{S}.$$

Flow chat of Q-learning



Q-learning can be executed with
pre-collected data with arbitrary stochastic policy

Q-learning

Theorem:

- Q-learning is one-step off-policy TD without explicitly containing any importance sampling (IS) ratio

Proof

- Behavior policy b is arbitrary stochastic policy
- (s, a) are known, and (s', a') are two random variables
- One-step off-policy TD estimation

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(\rho_{t+1:t} r + \gamma \rho_{t+1:t+1} Q(s',a') - Q(s,a)\big)$$

where $$\mathbb{E}_\pi\{r(s,a,s')\} = \mathbb{E}_b\{\rho_{t+1:t} r(s,a,s')\}$$

$$\rho_{t+1:t} = \frac{p(s'|s,a)}{p(s'|s,a)} = 1$$

$$\mathbb{E}_\pi\{Q(s',a')\} = \mathbb{E}_b\{\rho_{t+1:t+1} Q(s',a')\}$$

$$\rho_{t+1:t+1} = \frac{p(s'|s,a)\pi(a'|s')}{p(s'|s,a)b(a'|s')}$$

Proof (Continue)

- The key is to prove the equivalence of two expectations

$$\mathbb{E}_b\{\rho_{t+1:t}r + \gamma\rho_{t+1:t+1}Q(s',a')\} = \mathbb{E}_b\left\{r + \gamma\max_a Q(s',a)\right\}$$

- Use the fact that IS ratio depends on which variable is random

$$\begin{aligned}
&\mathbb{E}_b\{\rho_{t+1:t}r + \gamma\rho_{t+1:t+1}Q(s',a')\} \\
&= \mathbb{E}_{s'\sim\mathcal{P}}\left\{\frac{p(s'|s,a)}{p(s'|s,a)}r\right\} + \gamma\mathbb{E}_{s'\sim\mathcal{P},a'\sim b}\left\{\frac{p(s'|s,a)\pi(a'|s')}{p(s'|s,a)b(a'|s')}Q(s',a')\right\} \\
&= \mathbb{E}_{s'\sim\mathcal{P}}\{r\} + \gamma\mathbb{E}_{s'\sim\mathcal{P},a'\sim\pi}\{Q(s',a')\}
\end{aligned}$$

Because a' is from greedy search:

$$\begin{aligned}
&= \mathbb{E}_{s'\sim\mathcal{P}}\{r\} + \gamma\mathbb{E}_{s'\sim\mathcal{P}}\left\{\max_a Q(s',a)\right\} \\
&= \mathbb{E}_b\left\{r + \gamma\max_a Q(s',a)\right\}
\end{aligned}$$

Only one random variable is left, i.e., next state s'

Pseudocode of Q-learning

Hyperparameters: Discount factor γ, Learning rate α

Initialization: $Q(s, a) \leftarrow 0$, Behavior policy $b(a|s) \leftarrow 1/|\mathcal{A}|$

// Environment initialization

$s_0 \sim d_{\text{init}}(s)$

$s \leftarrow s_0$

Repeat until episode terminates (indexed with k)

//Sample action a under state s from behavior policy

$a \sim b(a|s)$

Rollout one step with action a, observing s' and r

//Do action-value update

$$Q(s, a) \leftarrow Q(s, a) + \alpha\left(r + \gamma \max_a Q(s', a) - Q(s, a)\right)$$

$s \leftarrow s'$

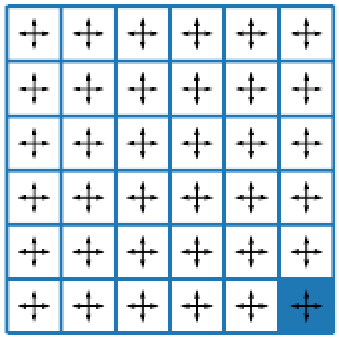
End

Calculate greedy policy $\pi(a|s)$ from $Q(s, a)$

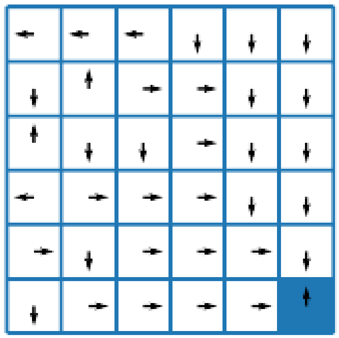
Q-learning

Example: Clean Robot

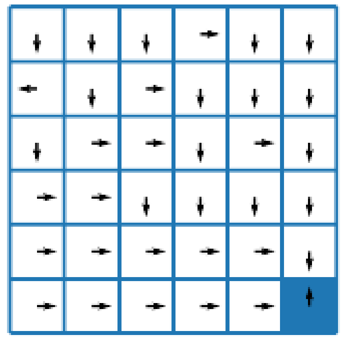
Improved Policy



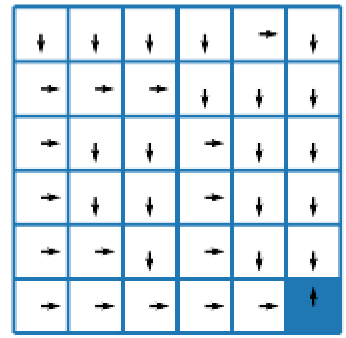
k=0



k=100

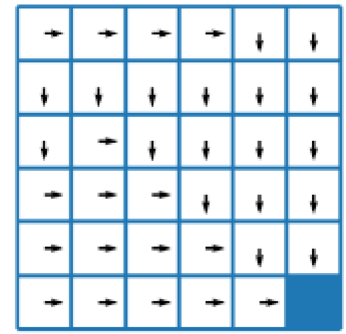


k=200



k=400

Final Policy



Estimated state value

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	0.0	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	0.0	0.0	

k=0

-2.8	-2.9	-3.2	-2.9	-2.7	-2.3
-2.8	-3.1	-2.9	-2.4	-2.0	-1.1
-3.0	-2.9	-2.5	-1.8	-0.8	0.4
-2.8	-2.5	-1.7	-0.8	1.2	1.9
-2.6	-1.8	-1.1	0.4	4.2	7.2
-2.1	-1.2	1.0	3.2	5.4	

k=100

-4.9	-5.0	-4.3	-3.3	-2.0	-1.4
-4.2	-4.2	-2.8	-1.4	-0.6	0.1
-4.1	-3.0	-1.5	0.3	0.9	2.2
-3.2	-1.9	-0.0	1.7	3.4	5.0
-2.5	-0.5	1.9	3.6	6.1	8.3
-1.9	0.1	2.8	5.6	7.6	

k=200

-4.1	-3.1	-2.0	-0.9	0.0	1.3
-3.1	-1.8	-0.8	0.6	1.9	2.5
-2.0	-0.7	0.2	1.9	3.4	4.7
-1.0	0.4	1.9	3.4	5.0	6.6
0.1	1.1	3.2	4.4	6.9	8.4
0.8	2.5	4.8	6.1	8.5	

k=400

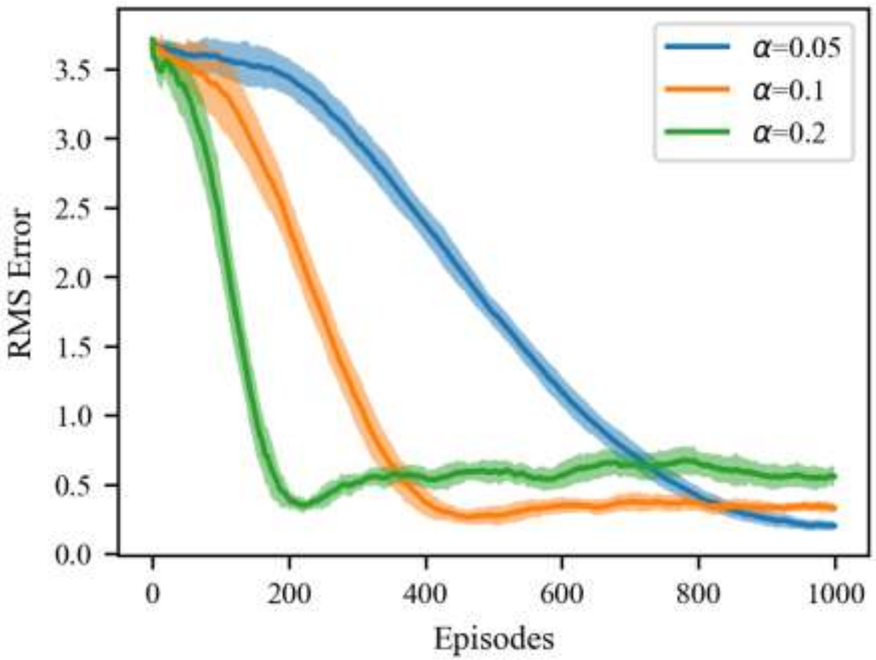
Final Value

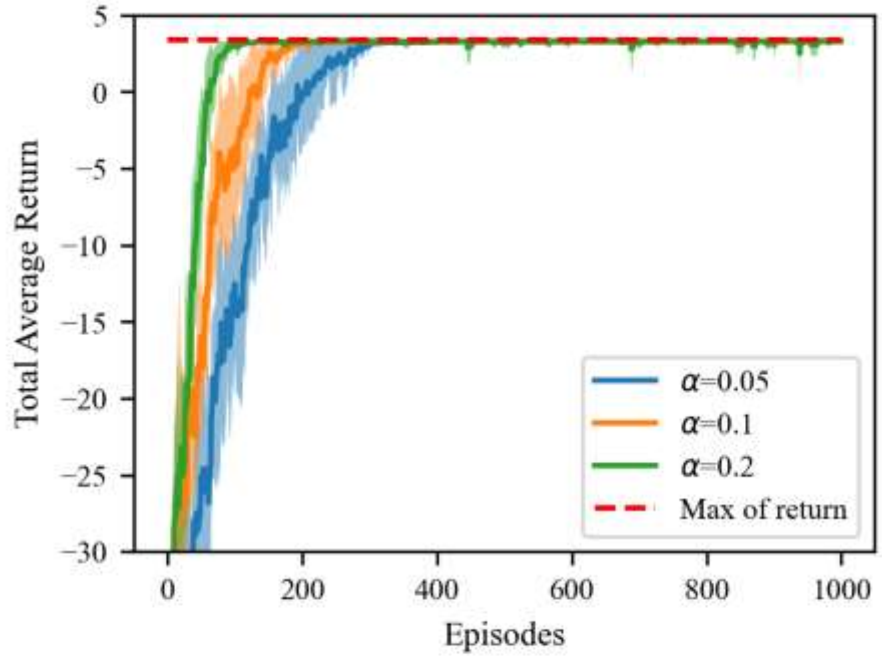
-3.1	-2.4	-1.3	-0.4	1.0	1.4
-2.3	-1.3	-0.1	0.8	2.2	3.2
-1.4	-0.1	1.1	2.5	3.6	4.9
-0.2	1.2	2.5	3.9	4.9	6.6
1.0	2.3	3.6	5.5	6.5	8.1
1.7	3.5	5.2	6.6	8.2	

Q-learning

Performance of Q-Learning

- Q-learning with different learning rate



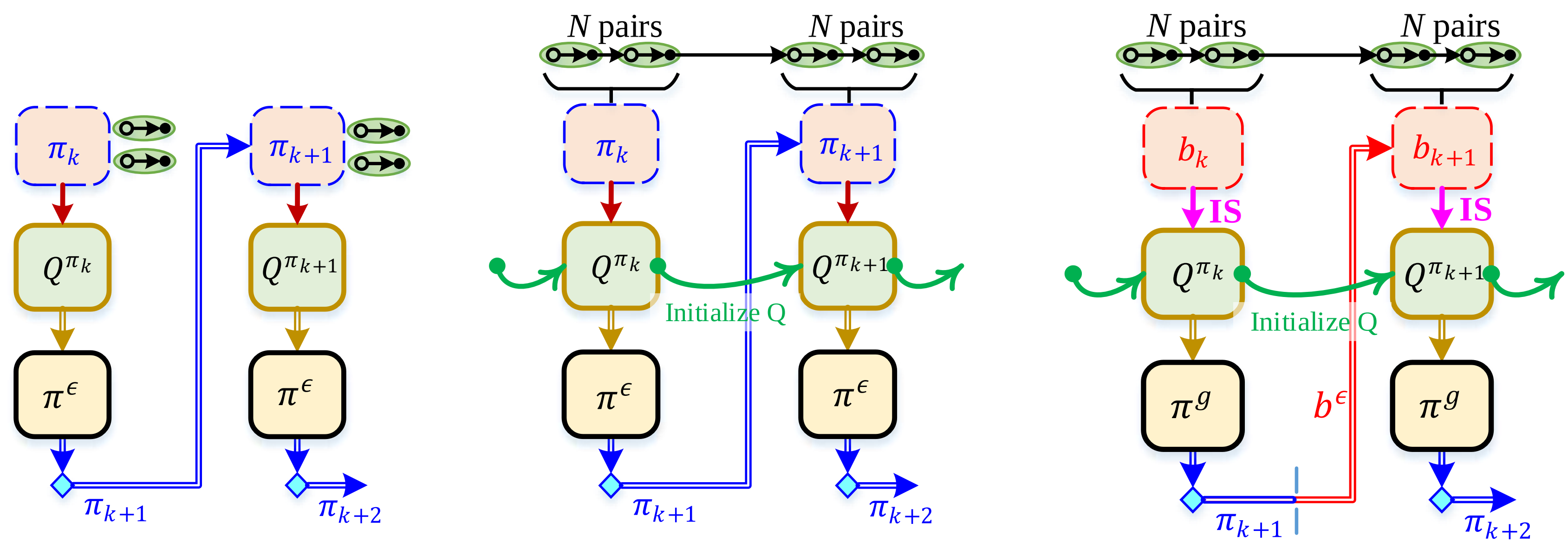


Large learning rate can accelerate convergence speed

Outline

Comparison of Model-free RL Algorithms

MC, Sarsa and Q-learning



- **MC**
- Use episodes in PEV
- On-policy/off-policy
- Often no initialization

- **Sarsa**
- Use N pairs/PEV
- On-policy
- With initialization

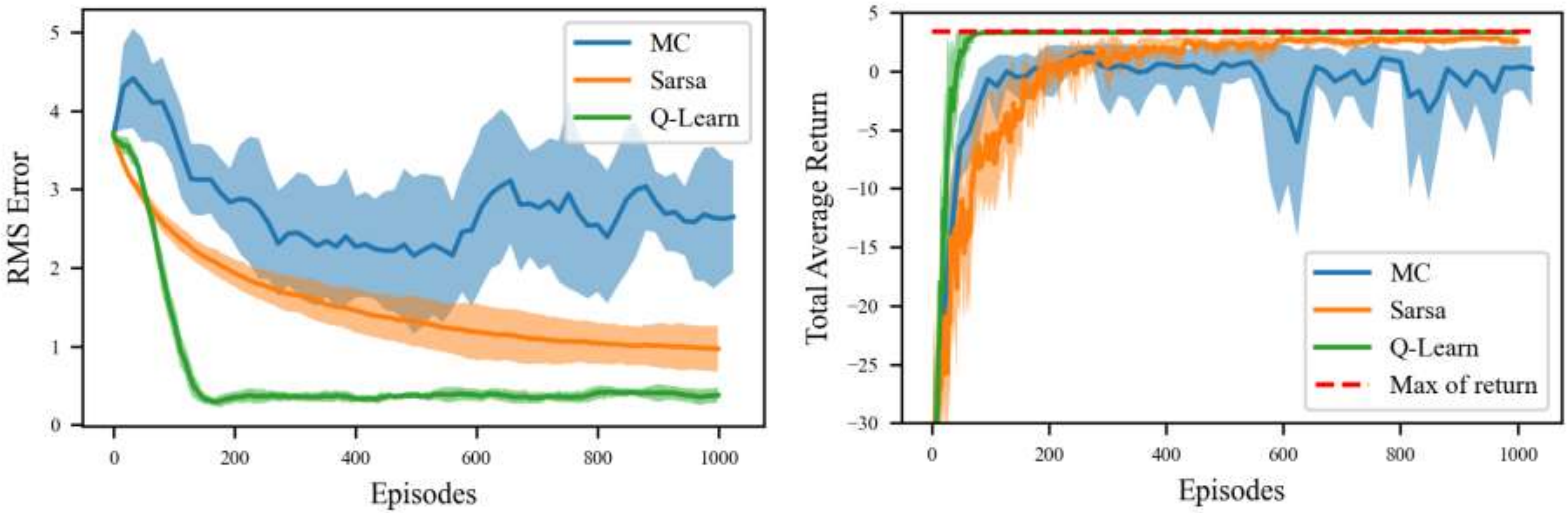
- **Off-policy TD**
- Q-learning = **1** pair/PEV
- Use pairs
- With initialization

?

If off-policy TD + N pairs/PEV, do we need IS ratio?

Comparison of RL Algorithms

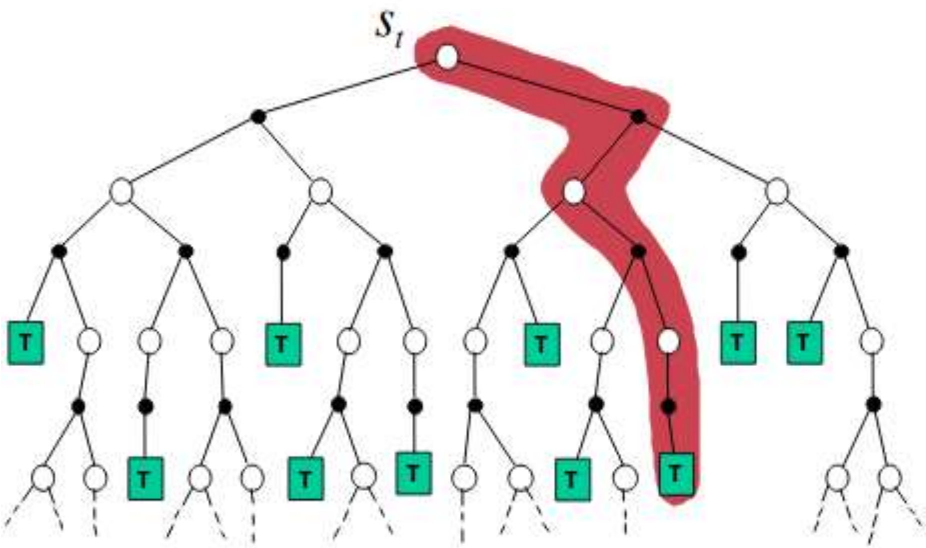
- **Compare of MC, Sarsa and Q-learning**



Mostly, Q-learning outperforms Sarsa and MC

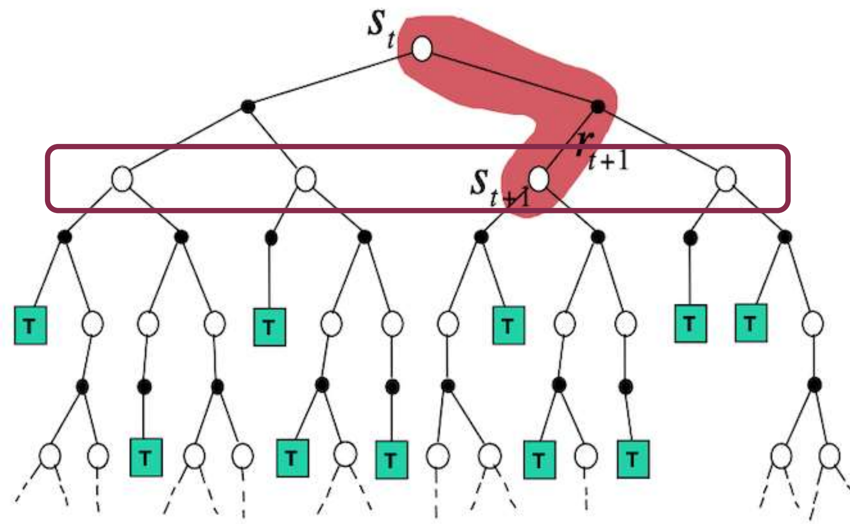
Backup Diagrams of TD and MC

Monte Carlo Estimation



Deep backup

Temporal Difference Estimation

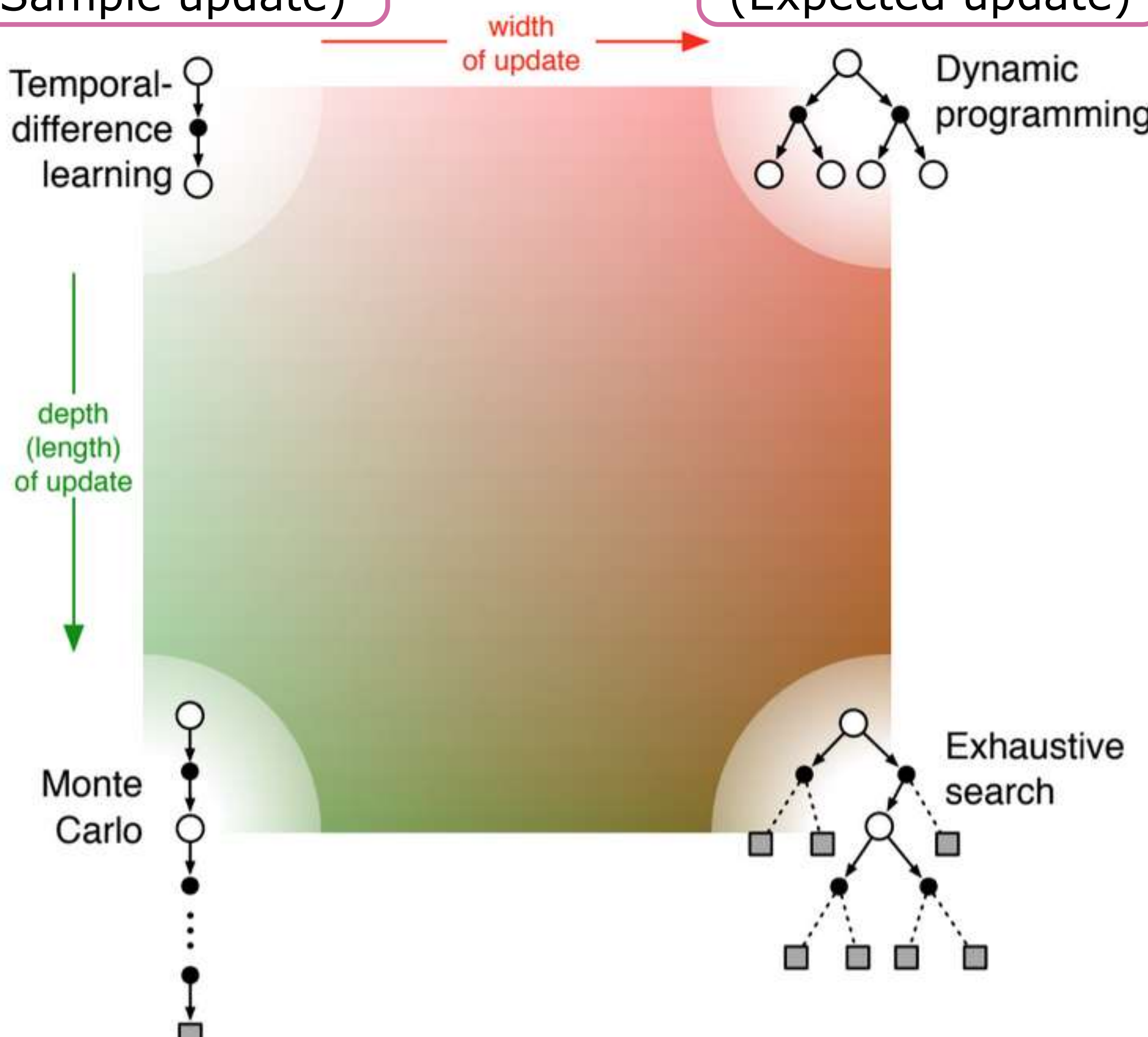


Shallow backup

Unification of TD and MC

Model-free
(Sample update)

Model-based
(Expected update)



Shallow
backup

?

Deep
backup

Unified View of TD and MC

N-step TD estimation

- Consider two sub-trajectories

$$\overbrace{r_t, r_{t+1}, r_{t+2}, \cdots, r_{t+n-1}}^{G_{t:t+n-1}}, \overbrace{r_{t+n}, r_{t+n+1}, \cdots}^{G_{t+n}}$$

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n G_{t+n}$$

- N-step self-consistency condition

$$v^\pi(s) = \mathbb{E}_\pi\{G_{t:t+n-1} + \gamma^n v^\pi(s_{t+n}) | s_t = s\}$$

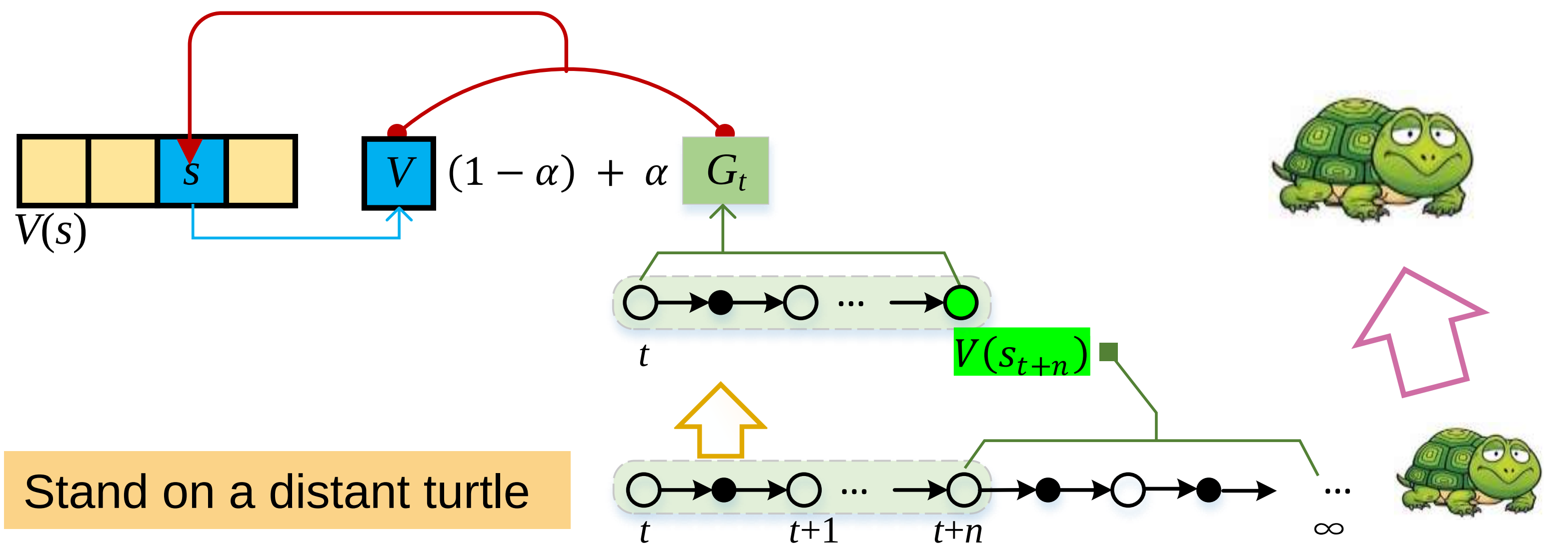
- Use bootstrapping to update state-value

$$\begin{aligned} V^\pi(s_t) &\leftarrow V^\pi(s_t) + \alpha\big(v^\pi(s) - V^\pi(s_t)\big) \\ &= V^\pi(s_t) + \alpha\big(G_{t:t+n-1} + \gamma^n V^\pi(s_{t+n}) - V^\pi(s_t)\big) \end{aligned}$$

N-step TD estimation

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\big(\underbrace{G_{t:t+n-1} + \gamma^n V^{\pi}(s_{t+n}) - V^{\pi}(s_t)}_{\text{N-step TD error}}\big)$$

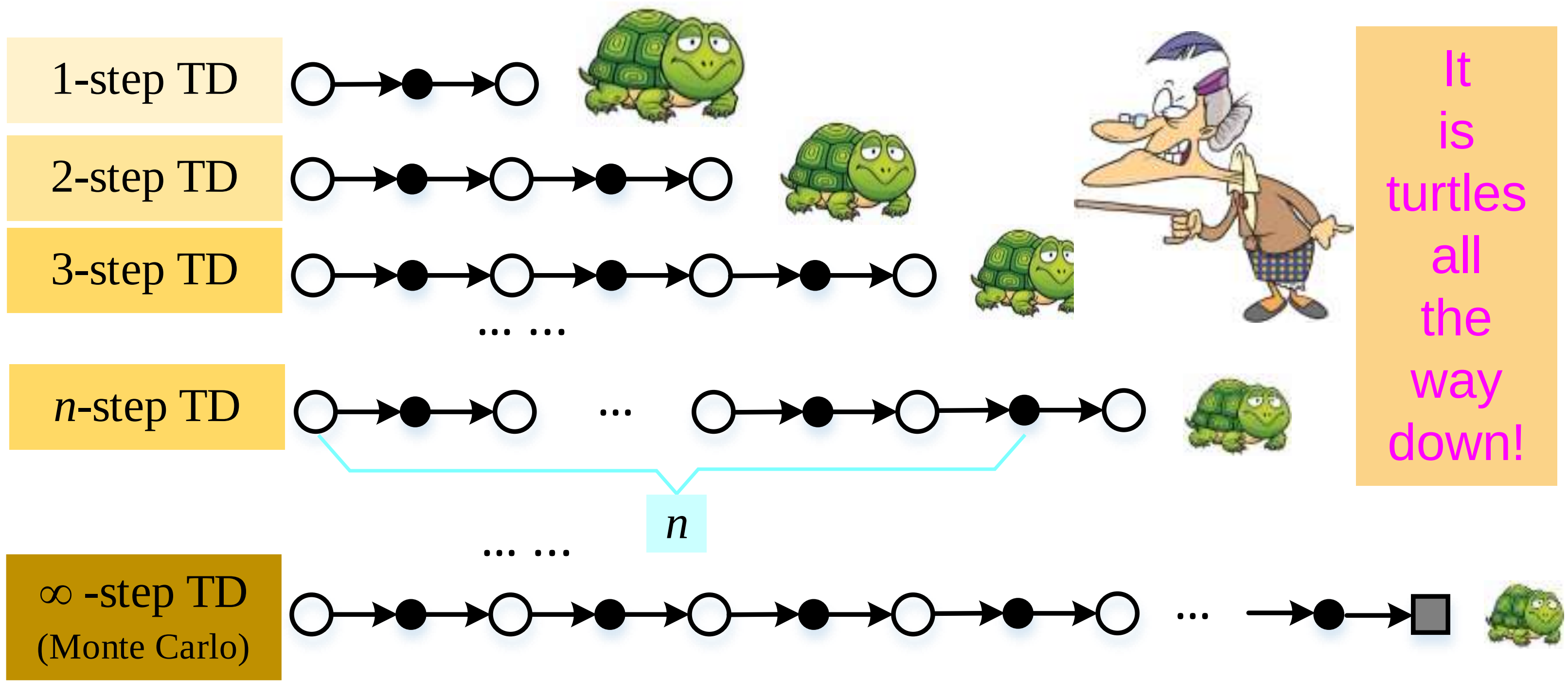
N-step TD error



Unified View of TD and MC

N-step TD estimation

- TD updates value based on one step reward
- MC updates value based on entire rewards
- N-step TD: more than one, but less than all of them



Benefit of n-step TD

- Sutton RS, Barto AG (2018)

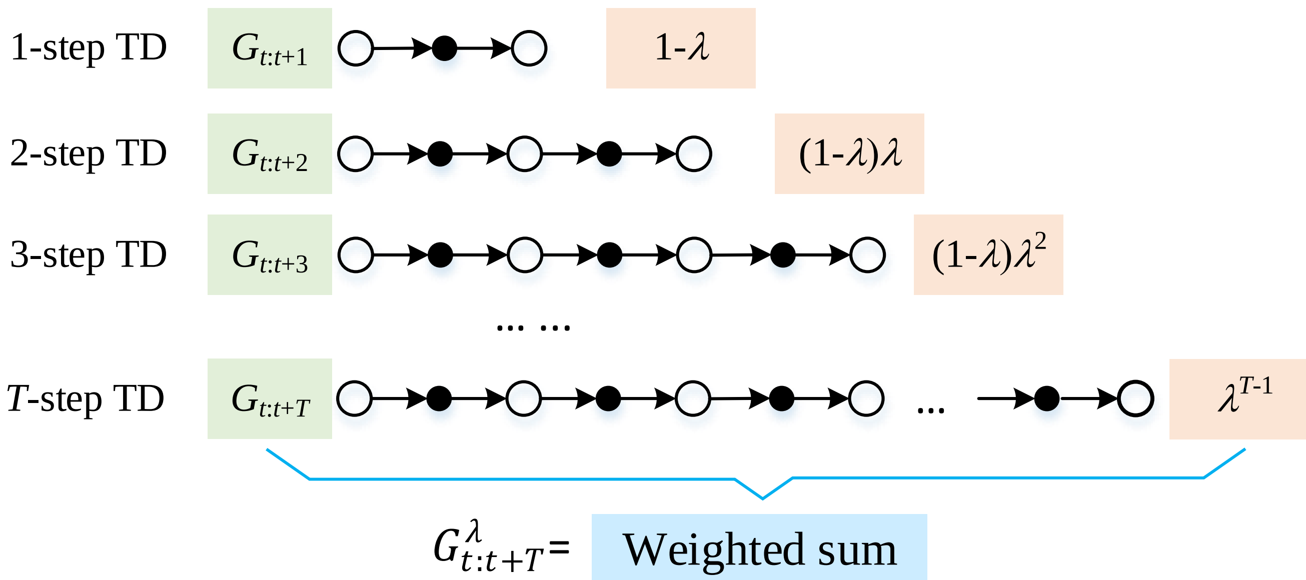
$$\max_s \left| \mathbb{E}_\pi\{G_{t:t+n-1} + \gamma^n V^\pi(s_{t+n}) | s_t = s\} - v^\pi(s) \right| \\ \leq \gamma^n \max_s \left| V^\pi(s) - v^\pi(s) \right|$$

- An important property of n-step TD is that their expectation is guaranteed to be a better estimate of true state-value

- The worst-case error of the expected n-step TD is less than or equal to γ^n times that of an estimated state-value

TD-Lambda estimation

- Singh and Sutton (1996)
- Eligibility trace is a way of efficiently computing TD-lambda



$T \to \infty$

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1}\left(G_{t:t+n-1} + \gamma^n V^{\pi}(s_{t+n})\right)$$

The End!