



第七章 强化学习III—无模型值方法

Chao Yu (余超)

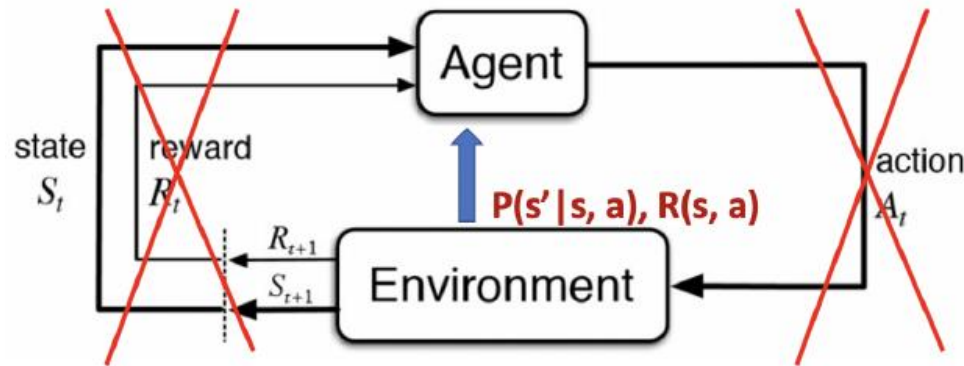
School of Computer Science and Engineering
Sun Yat-Sen University

[based on David Silver and Sergey Levine's course]

- Last lecture:
 - MDP
 - policy evaluation
 - policy iteration and value iteration for solving a known MDP
- This lecture:
 - Model-free prediction: Estimate value function of an unknown MDP
 - Model-free control: Optimize value function of an unknown MDP

RL with knowing how the world works

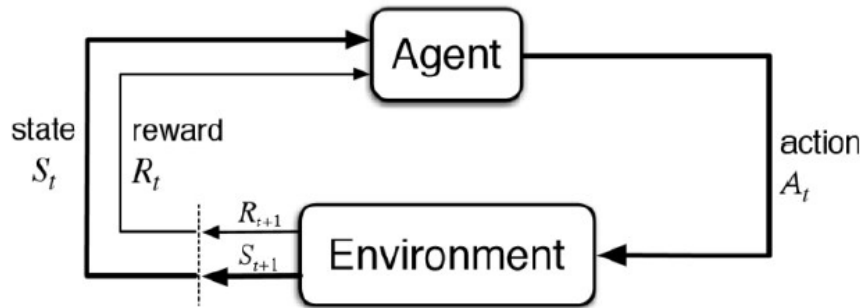
- Both of the policy iteration and value iteration assume the direct access to the dynamics and rewards of the environment



- In a lot of real-world problems, MDP model is either unknown or known by too big or too complex to use
 - Atari Game, Game of Go, Helicopter, Portfolio management, etc

Model-free RL: Learning by interaction

- ❑ Model-free RL can solve the problems through interaction with the environment



- ❑ No more direct access to the known transition dynamics and reward function
- ❑ Trajectories/episodes are collected by the agent's interaction with the environment
- ❑ Each trajectory/episode contains $\{S_1, A_1, R_1, S_2, A_2, R_2, \dots, S_T, A_T, R_T\}$

- ❑ Model-free prediction: policy evaluation without the access to the model
- ❑ Estimating the expected return of a particular policy if we don't have access to the MDP models
 - ❑ Monte Carlo policy evaluation
 - ❑ Temporal Difference (TD) learning

Monte-Carlo Policy Evaluation

- ❑ Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$
- ❑ $v^\pi(s) = \mathbb{E}_{\tau \sim \pi}[G_t | s_t = s]$ thus expectation over trajectories τ generated by following π
- ❑ MC simulation: we can simply sample a lot of trajectories, compute the actual returns for all the trajectories, then average them
- ❑ MC policy evaluation uses empirical mean return instead of expected return
- ❑ MC does not require MDP dynamics/rewards, no bootstrapping, and does not assume state is Markov.
- ❑ Only applied to episodic MDPs (each episode terminates)

Monte-Carlo Policy Evaluation



□ To evaluate state $v(s)$

- ① Every time-step t that state s is visited in an episode,
- ② Increment counter $N(s) \leftarrow N(s) + 1$
- ③ Increment total return $S(s) \leftarrow S(s) + G_t$
- ④ Value is estimated by mean return $v(s) = S(s)/N(s)$

□ By law of large numbers, $v(s) \rightarrow v^\pi(s)$ as $N(s) \rightarrow \infty$

Incremental MC Updates

- Mean from the average of samples x_1, x_2, \dots

$$\begin{aligned}\mu_t &= \frac{1}{t} \sum_{j=1}^t x_j \\ &= \frac{1}{t} \left(x_t + \sum_{j=1}^{t-1} x_j \right) \\ &= \frac{1}{t} (x_t + (t-1)\mu_{t-1}) \\ &= \mu_{t-1} + \frac{1}{t} (x_t - \mu_{t-1})\end{aligned}$$

- Collect one episode $(S_1, A_1, R_1, \dots, S_t)$

- For each state s_t with computed return G_t

$$N(S_t) \leftarrow N(S_t) + 1$$

$$v(S_t) \leftarrow v(S_t) + \frac{1}{N(S_t)} (G_t - v(S_t))$$

- Or use a running mean (old episodes are forgotten). Good for non-stationary problems.



$$v(S_t) \leftarrow v(S_t) + \alpha (G_t - v(S_t))$$

Temporal-Difference (TD) Learning

- ❑ TD methods learn directly from episodes of experience
- ❑ TD is model-free: no knowledge of MDP transitions/rewards
- ❑ TD learns from incomplete episodes, by bootstrapping
- ❑ Objective: learn v_π online from experience under policy π
- ❑ Simplest TD algorithm: TD(0)

① Update $v(S_t)$ toward estimated return $R_{t+1} + \gamma v(S_{t+1})$

$$v(S_t) \leftarrow v(S_t) + \alpha (R_{t+1} + \gamma v(S_{t+1}) - v(S_t))$$

$\delta_t = R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$  TD error  TD target

- ❑ Comparison: Incremental Monte-Carlo

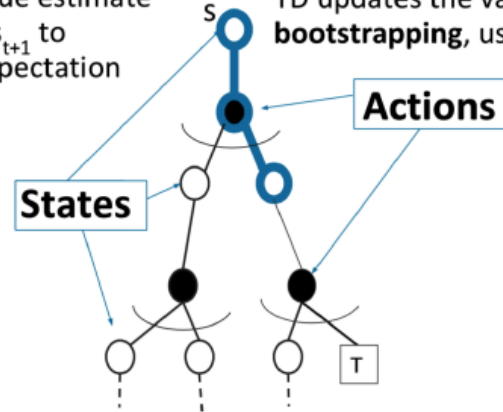
① Update $v(S_t)$ toward actual return G_t given an episode i

$$v(S_t) \leftarrow v(S_t) + \alpha (G_{i,t} - v(S_t))$$

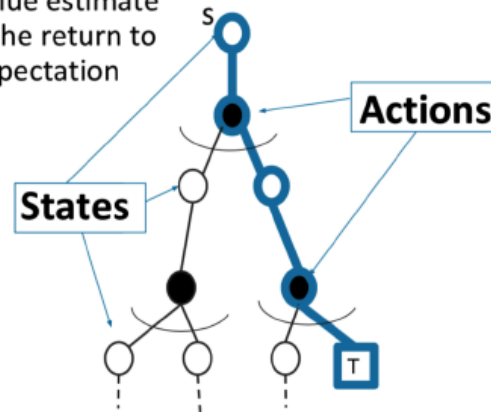
Advantages of TD over MC

TD updates the value estimate using a **sample** of s_{t+1} to approximate an expectation

TD updates the value estimate by **bootstrapping**, uses estimate of $V(s_{t+1})$



MC updates the value estimate using a **sample** of the return to approximate an expectation



⌋ = Expectation

⌈ T ⌋ = Terminal state

Comparison of TD and MC



- ❑ TD can learn online after every step
- ❑ MC must wait until end of episode before return is known

- ❑ TD can learn from incomplete sequences
- ❑ MC can only learn from complete sequences

- ❑ TD works in continuing (non-terminating) environments
- ❑ MC only works for episodic (terminating) environments

- ❑ TD exploits Markov property, more efficient in Markov environments
- ❑ MC does not exploit Markov property, more effective in non-Markov environments

Comparison of TD and MC



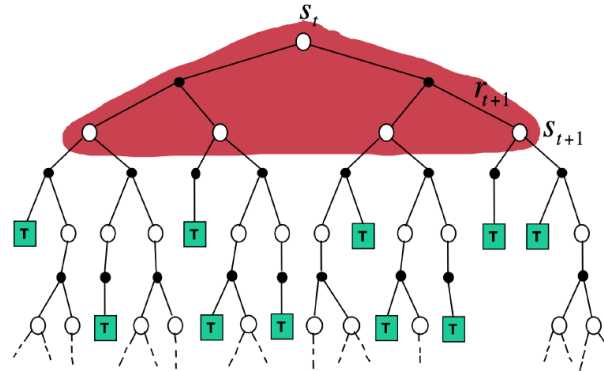
- ❑ MC has high variance, zero bias
 - ❑ Good convergence properties
 - ❑ (even with function approximation)
 - ❑ Not very sensitive to initial value
 - ❑ Very simple to understand and use

- ❑ TD has low variance, some bias
 - ❑ Usually more efficient than MC
 - ❑ TD(0) converges to $V_{\pi}(s)$
 - ❑ (but not always with function approximation)
 - ❑ More sensitive to initial value

- ❑ Bootstrapping: update involves an estimate
 - ❑ MC does not bootstrap
 - ❑ DP bootstraps
 - ❑ TD bootstraps

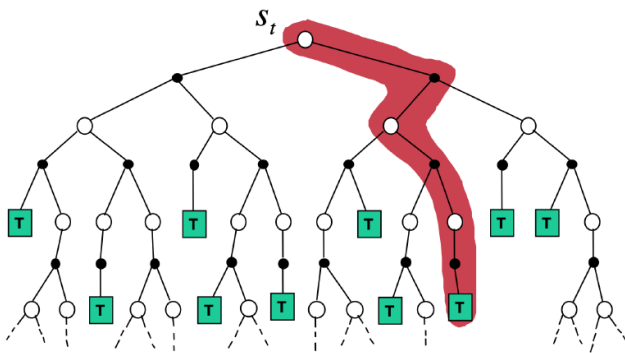
- ❑ Sampling: update samples an expectation
 - ❑ MC samples
 - ❑ DP does not sample
 - ❑ TD samples

$$v(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma v(S_{t+1})]$$



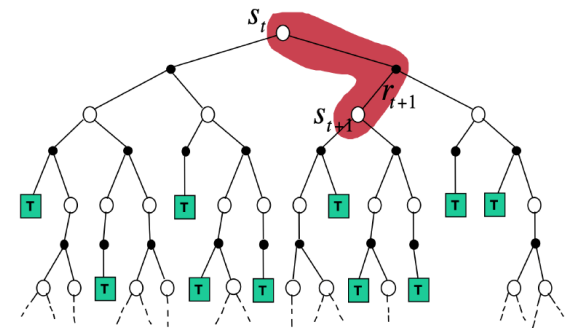
Dynamic Programming Backup

$$v(S_t) \leftarrow v(S_t) + \alpha(G_t - v(S_t))$$



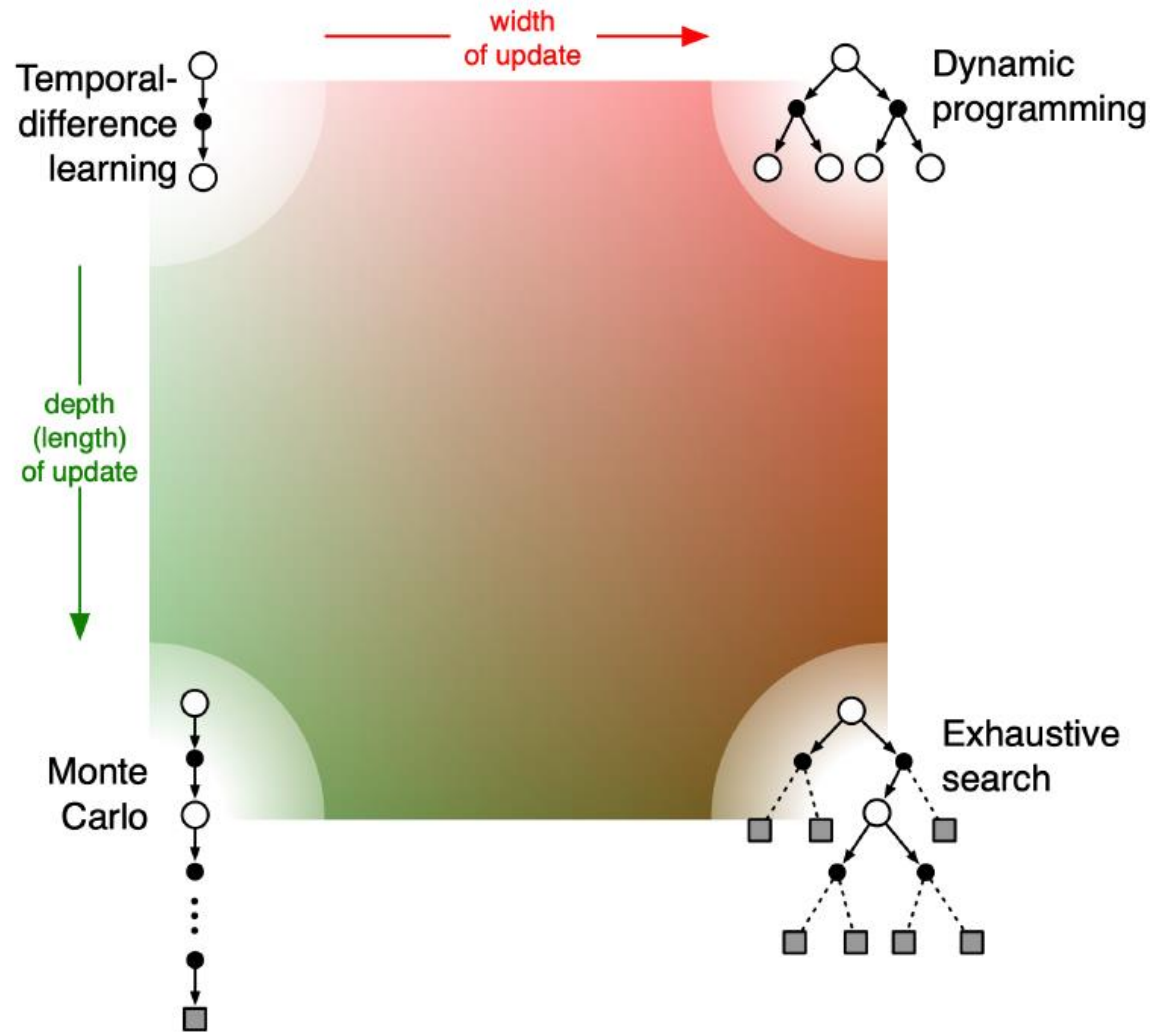
Monte-Carlo Backup

$$TD(0) : v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(s_{t+1}) - v(S_t))$$



Temporal-Difference Backup

Unified View of RL



- ❑ Model-Free Reinforcement Learning
 - ❑ Model-free prediction
 - ❑ Estimate the value function of an unknown MDP
 - ❑ Model-free control
 - ❑ Monte-Carlo control
 - ❑ Temporal Difference (TD) control
 - ❑ Off-Policy Learning
- ❑ Optimise the value function of an unknown MDP

Uses of Model-Free Control

Some example problems that can be modelled as MDPs

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Aeroplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein Folding
- Robot walking
- Game of Go

For most of these problems, either:

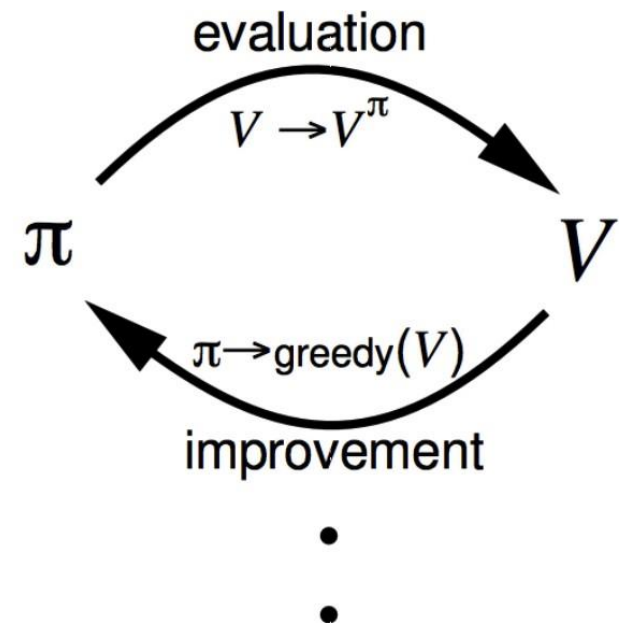
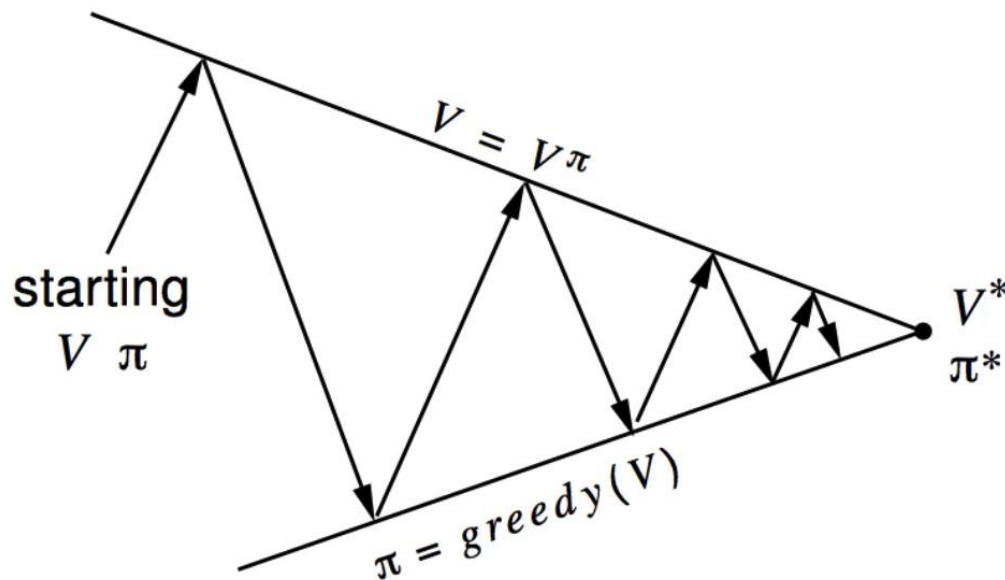
- MDP model is unknown, but experience can be sampled
- MDP model is known, but is too big to use, except by samples

Model-free control can solve these problems

Policy Iteration

- Iteration through the two steps
 - Evaluate the policy π (computing v given current π)
 - Improve the policy by acting greedily with respect to v_π

$$\pi' = \text{greedy}(v_\pi)$$



Policy Iteration for a Known MDP



- Compute the state-action value of a policy π :

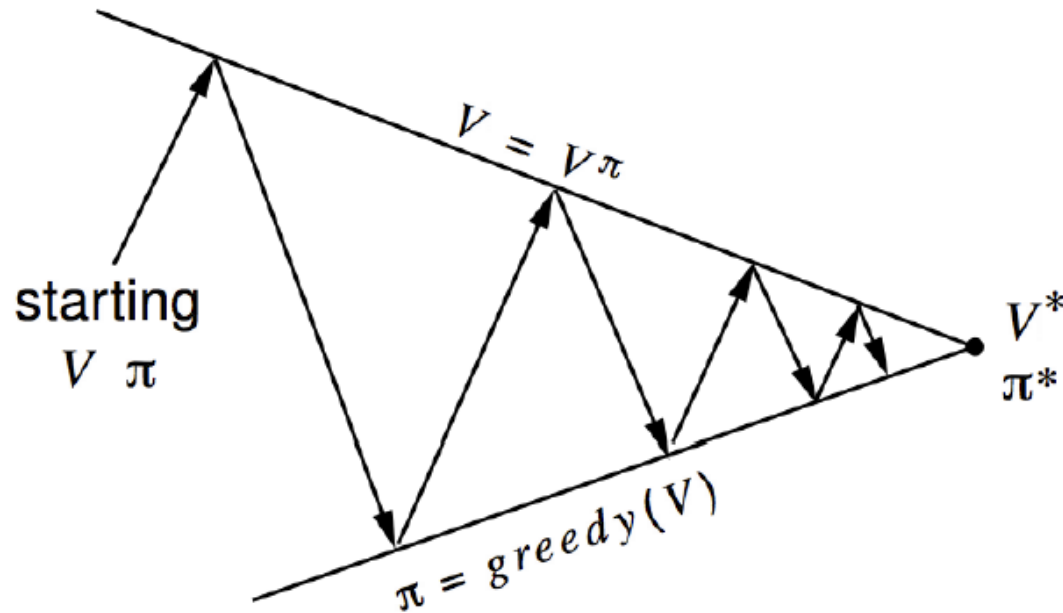
$$q_{\pi_i}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v_{\pi_i}(s')$$

- Compute new policy π_{i+1} for all $s \in S$ following

$$\pi_{i+1}(s) = \arg \max_a q_{\pi_i}(s, a)$$

- Problem: what to do if there is neither $R(s, a)$ nor $P(s'|s, a)$ known/available

General PI With Monte-Carlo Evaluation



Policy evaluation Monte-Carlo policy evaluation, $V = v_\pi$?

Policy improvement Greedy policy improvement?

Monte Carlo with ϵ – *Greedy* Exploration



- ϵ – *greedy* Exploration: Ensuring continual exploration
 - All actions are tried with non-zero probability
 - With probability $1 - \epsilon$ choose the greedy action
 - With probability ϵ choose an action at random

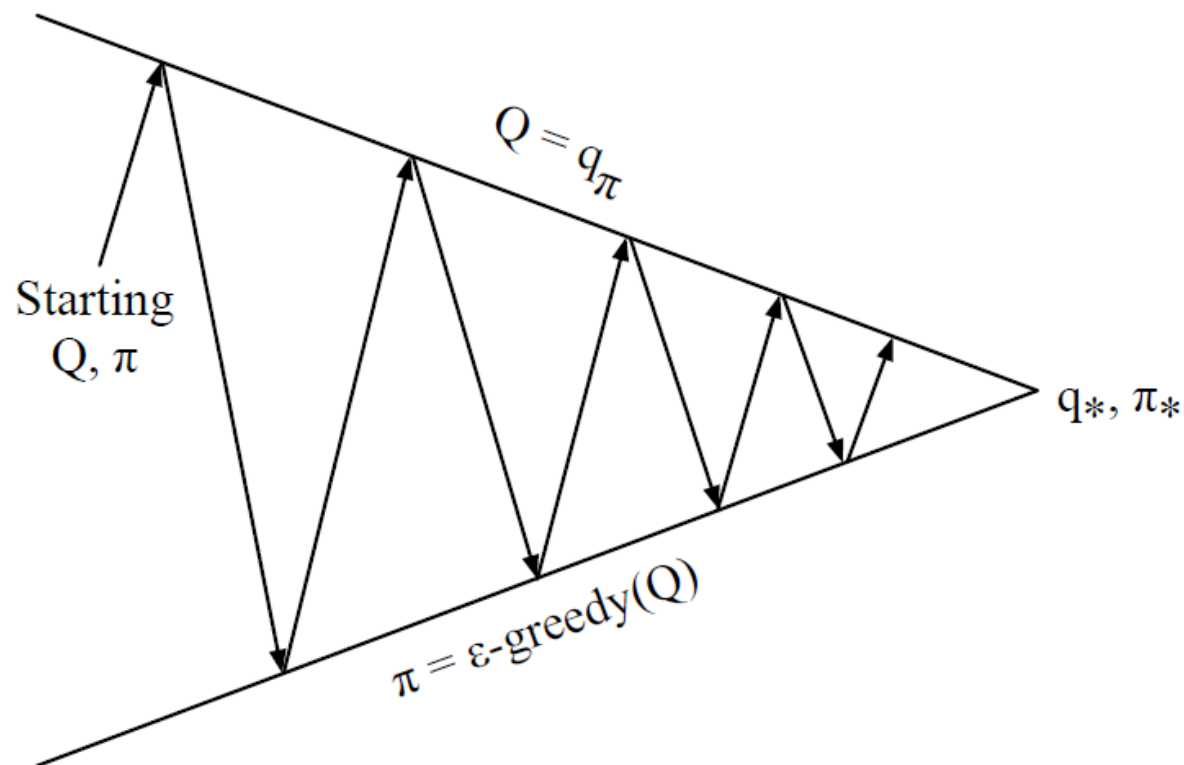
$$\pi(a|s) = \begin{cases} \epsilon/|\mathcal{A}| + 1 - \epsilon & \text{if } a^* = \arg \max_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/|\mathcal{A}| & \text{otherwise} \end{cases}$$

□ **Policy improvement theorem:** For any policy π , the ϵ – greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\ &= \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_a q_\pi(s, a) \\ &\geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{|\mathcal{A}|}}{1 - \epsilon} q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s) \end{aligned}$$

Therefore, $v_{\pi'}(s) \geq v_\pi(s)$ from the policy improvement theorem

Monte-Carlo Policy Iteration



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

Policy improvement ϵ -greedy policy improvement

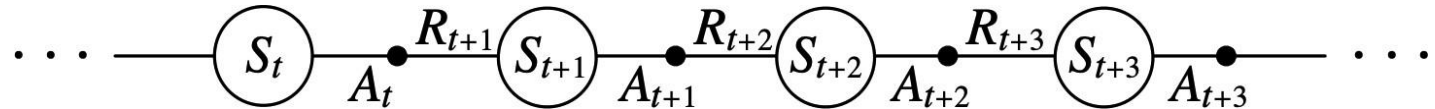
MC VS. TD for Prediction and Control



- ❑ Temporal-difference(TD) learning has several advantages over Monte-Carlo(MC)
 - ❑ Lower variance
 - ❑ Online
 - ❑ Incomplete sequences

- ❑ So we can use TD instead of MC in our control loop
 - ❑ Apply TD to $Q(S, A)$
 - ❑ Use ϵ – *greedy* policy improvement
 - ❑ Update every time-step rather than at the end of one episode

- An episode consists of an alternating sequence of states and state-action pairs:



- ϵ - Greedy policy for one step, then bootstrap the action value function:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right]$$

- The update is done after every transition from a nonterminal state S_t
- TD target: $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$

- We allow both behavior and target policies to improve

- The target policy π is **greedy** on $Q(s, a)$

$$\pi(S_{t+1}) = \arg \max_{a'} Q(S_{t+1}, a')$$

- The behavior policy μ could be totally random, but we let it improve following **ϵ – greedy** on $Q(s, a)$

- Thus Q-learning target

$$\begin{aligned} R_{t+1} + \gamma Q(S_{t+1}, A') &= R_{t+1} + \gamma Q(S_{t+1}, \arg \max_{a'} Q(S_{t+1}, a')) \\ &= R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') \end{aligned}$$

- Thus the Q-learning update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Comparison of Sarsa and Q-learning

□ Sarsa: On-Policy TD control

Choose action A_t from S_t using policy derived from Q with $\epsilon - greedy$

Take action A_t , observe R_{t+1} and S_{t+1}

Choose action A_{t+1} from S_{t+1} using policy derived from Q with $\epsilon - greedy$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

□ Q-learning: Off-Policy TD control

Choose action A_t from S_t using policy derived from Q with $\epsilon - greedy$

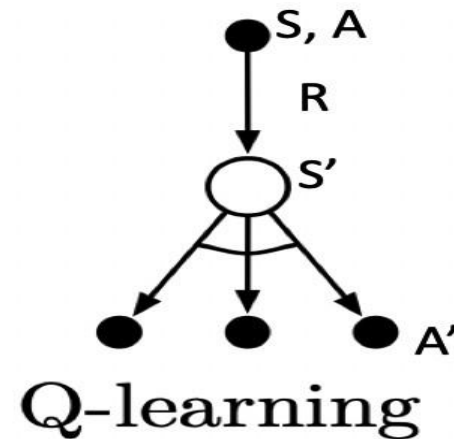
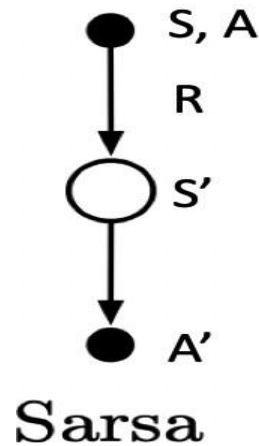
Take action A_t , observe R_{t+1} and S_{t+1}

Then ‘imagine’ A_{t+1} as $\operatorname{argmax}_a Q(S_{t+1}, a)$ in the update target

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Comparison of Sarsa and Q-learning

□ Backup diagram for Sarsa and Q-learning



- In Sarsa, A and A' are sampled from the same policy so it is on-policy
- In Q-learning, A and A' are from different policies, with A being more exploratory and A' determined directly by the max operator

Comparison of Sarsa and Q-learning

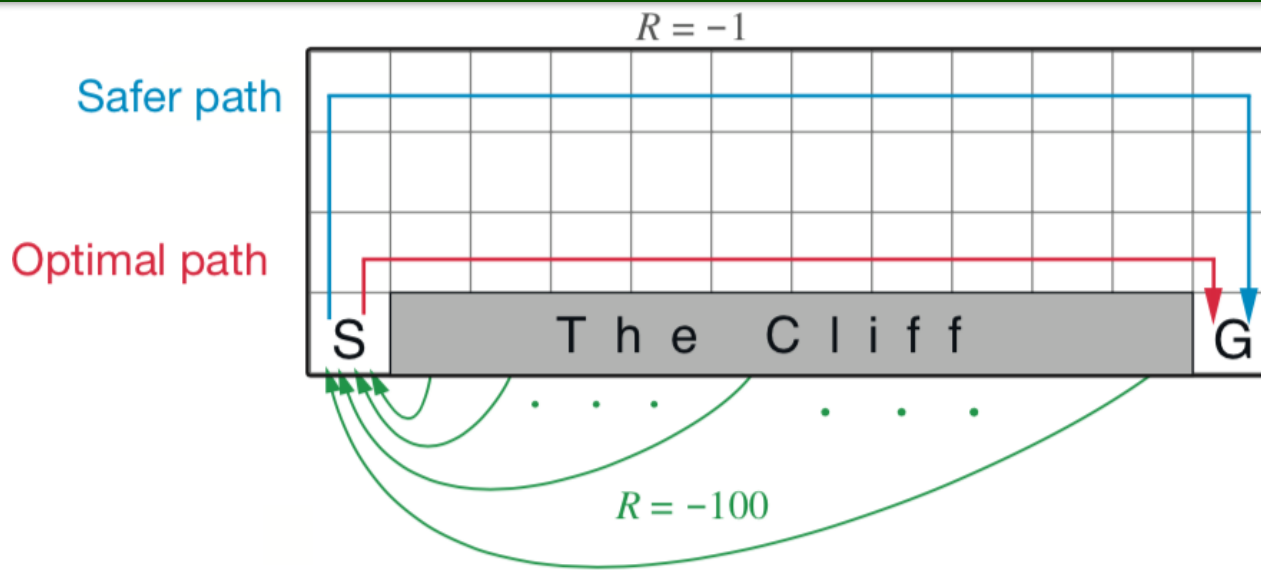
□ Sarsa

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
Repeat (for each episode):
 Initialize S
 Choose A from S using policy derived from Q (e.g., ϵ -greedy)
 Repeat (for each step of episode):
 Take action A , observe R, S'
 Choose A' from S' using policy derived from Q (e.g., ϵ -greedy)
 $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$
 $S \leftarrow S'; A \leftarrow A';$
 until S is terminal

□ Q learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
Repeat (for each episode):
 Initialize S
 Repeat (for each step of episode):
 Choose A from S using policy derived from Q (e.g., ϵ -greedy)
 Take action A , observe R, S'
 $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
 $S \leftarrow S';$
 until S is terminal

Example on Cliff Walk



0	0	0	0	R	R	R	R	R	R	R	R
R	R	R	R	R	0	0	0	0	0	0	R
R	0	0	0	0	0	0	0	0	0	0	R
R	*	*	*	*	*	*	*	*	*	*	G

Result of Sarsa

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R
R	*	*	*	*	*	*	*	*	*	*	G

Result of Q-Learning



On-line performance of Q-learning is worse than that of Sarsa

Summary of DP and TD

Expected Update (DP)	Sample Update (TD)
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}[R + \gamma V(S') s]$	TD Learning $V(S) \leftarrow^{\alpha} R + \gamma V(S')$
Q-Policy Iteration $Q(S, A) \leftarrow \mathbb{E}[R + \gamma Q(S', A') s, a]$	Sarsa $Q(S, A) \leftarrow^{\alpha} R + \gamma Q(S', A')$
Q-Value Iteration $Q(S, A) \leftarrow \mathbb{E}[R + \gamma \max_{a' \in \mathcal{A}} Q(S', A') s, a]$	Q-Learning $Q(S, A) \leftarrow^{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S', a')$

where $x \leftarrow^{\alpha} y$ is defined as $x \leftarrow x + \alpha(y - x)$