Introduction to Reinforcement Learning based Control and Deep Reinforcement Learning based control

Associate Professor
(Maitre de Conférences)
Automatic Control, Reliability of Systems
mayank-shekhar.jha{at}univ-lorraine.fr

Research
Centre de Recherche en Automatique de Nancy
(CRAN) UMR 7039,
Faculté des Sciences et Technologies
Boulevard des Aiguillettes - BP 70239 - Bât.
1er cycle 54506 Vandoeuvre-lès-Nancy
Cedex France

Teaching
Polytech Nancy (ESSTIN),
2 Rue Jean Lamour 54509
Vandoeuvre-lès-Nancy,
Cedex France

Email: mayank-shekhar.jha [at] univ-lorraine.fr

Introduction

Markov Decision process

Dynamic Programming for planning and control

Model Free Prediction and Control (SARSA and Q-learning)

Model-free Prediction

Temporal Difference (TD)

POLYTECH NANCY
UNIVERSITÉ DE LORRAINE
CRAN

JHA Mayank , Email: mayank-shekhar.jha [at] univ-lorraine.fr

So far:

- Definition, setting of RL, MDP,
- Solving a known MDP

Here

- Model-free Prediction
- Estimate Value Function of unknown MDP

Next:

- Model-free control
- Optimisation of unknown MDP
- Approximate DP : Deep Learing + Reinforcement Learning.

Temporal Difference (TD)

- learns directly from 'episodes' of experiences
- TD is model-free : no knowledge of MDP transitions / rewards required
- TD: learns by bootstrapping (update a guess using a guess)
- TD: can learn from incomplete episodes.

Goal: learn v_π online from experience under policy π

Temporal Difference TD(0) algorithm

- Update value $V(S_t)$ towards estimated return $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha \left(\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{estimated return}} - \underbrace{V(S_t)}_{\text{present value.}} \right)$$

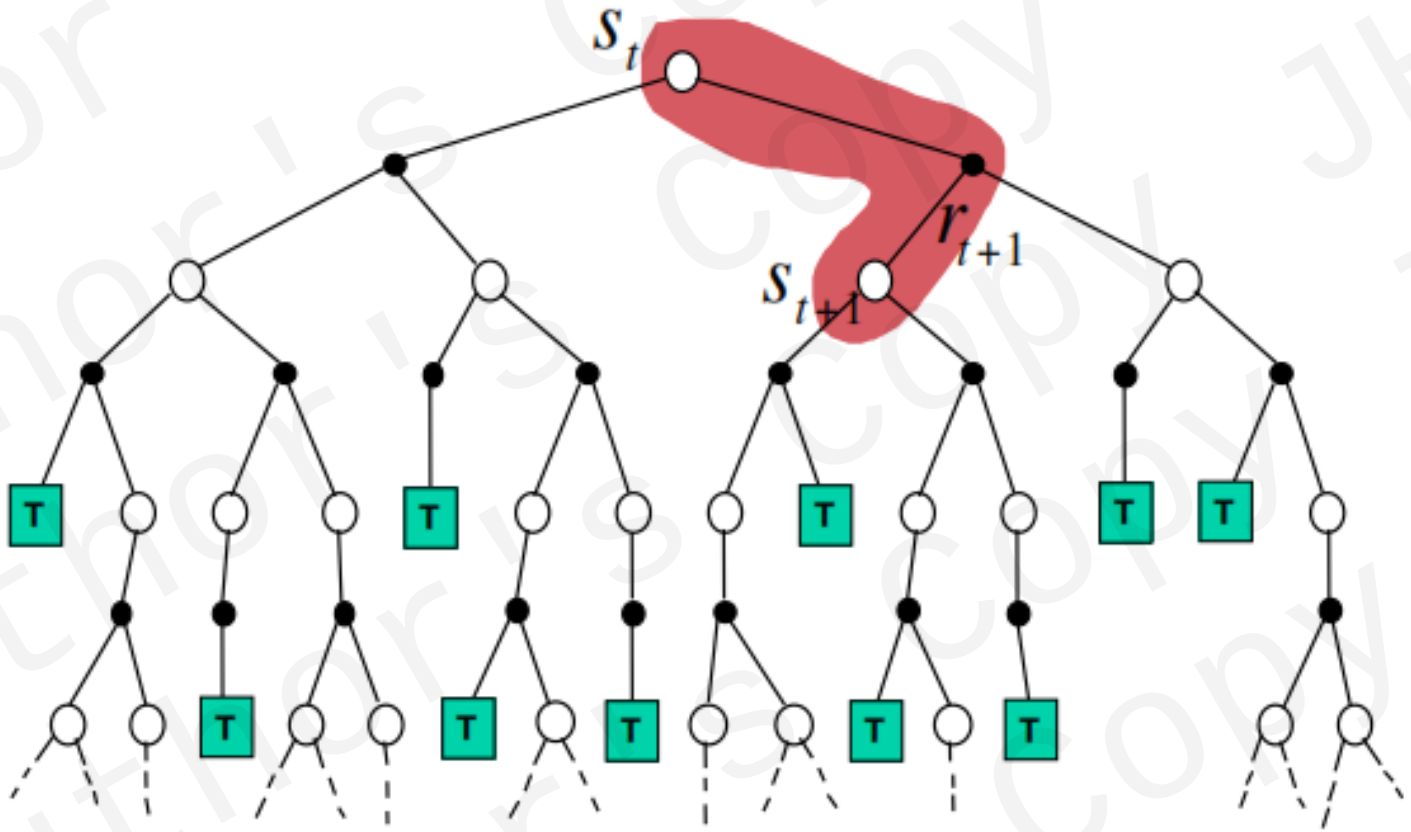
difference between some estimate of return and actual return.

TD target : $R_{t+1} + \gamma V(S_{t+1})$

TD error : $\delta_t = \text{TD target} - V(S_t)$

↑ actual value

TD(0) based back up



Monte Carlo (MC) learning

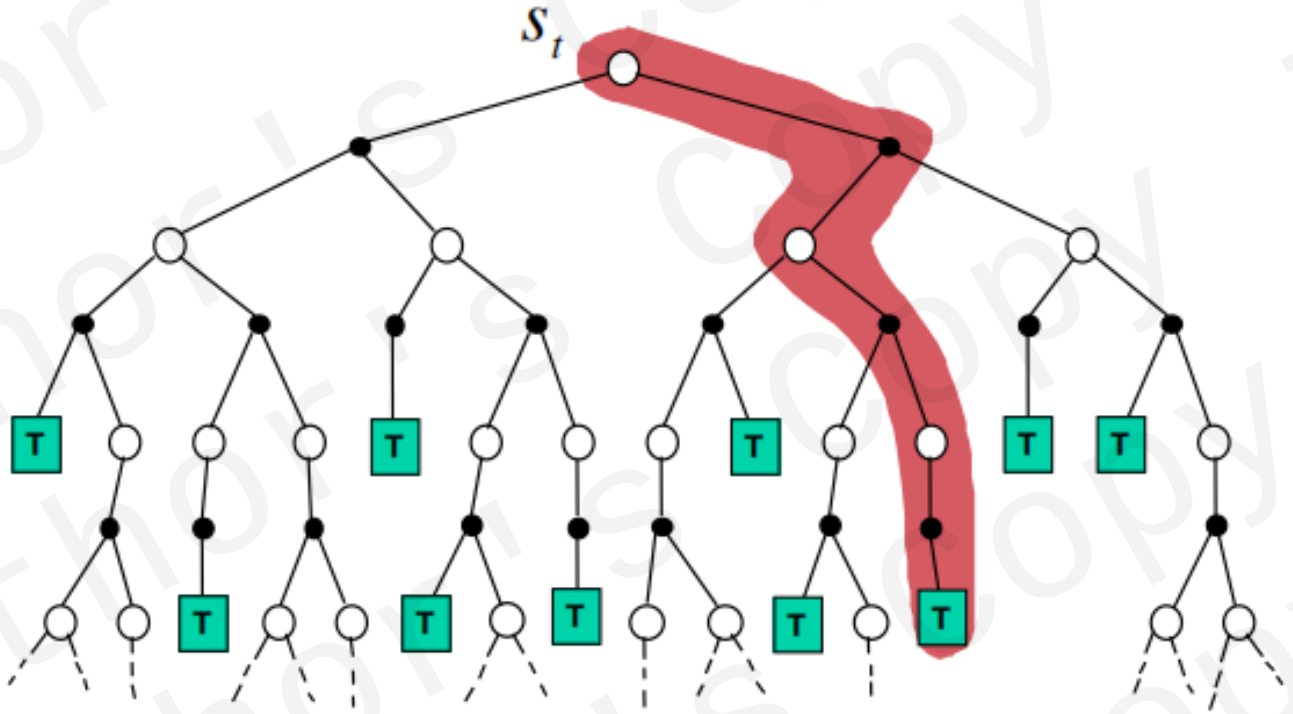
- State 's' is evaluated based on every state visited in an episode
- Update $V(S_t)$ towards actual (not estimated) return G_t

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

Limitation

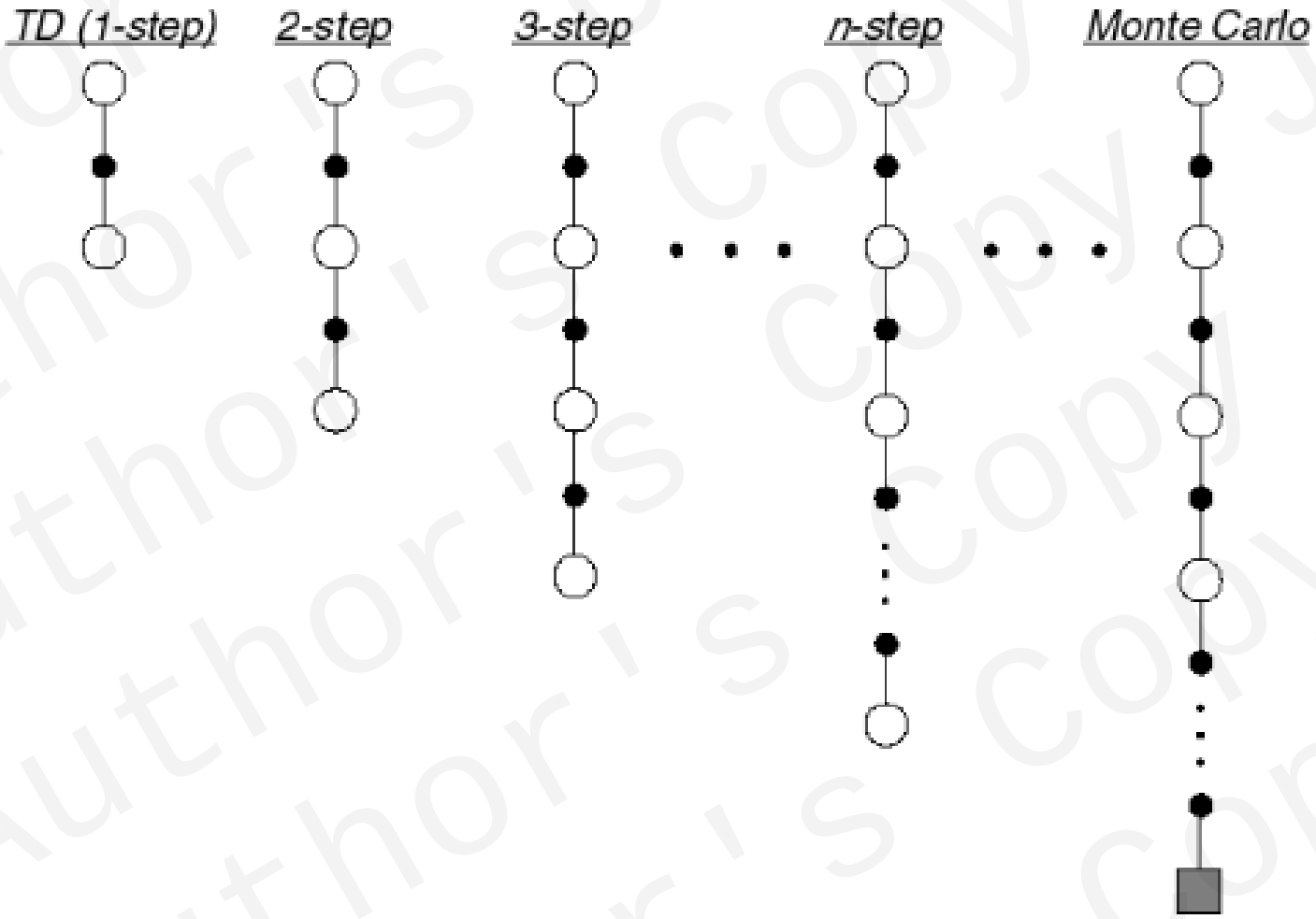
- Update only after episode end.
- Cannot learn on the run!!

Monte Carlo based back-up.



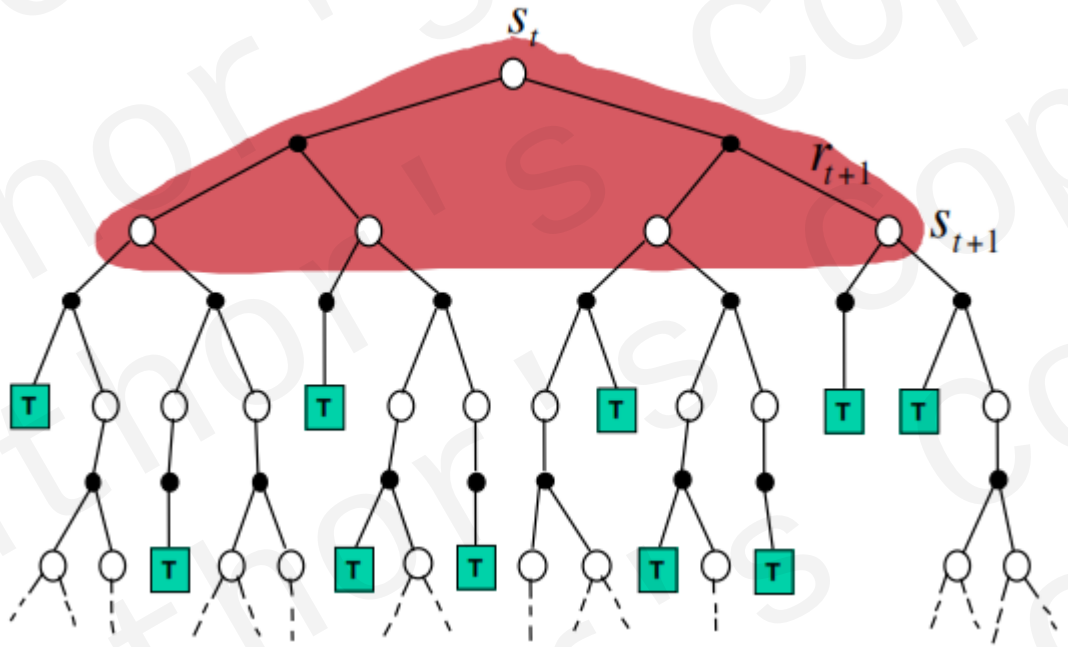
Difference TD and MC

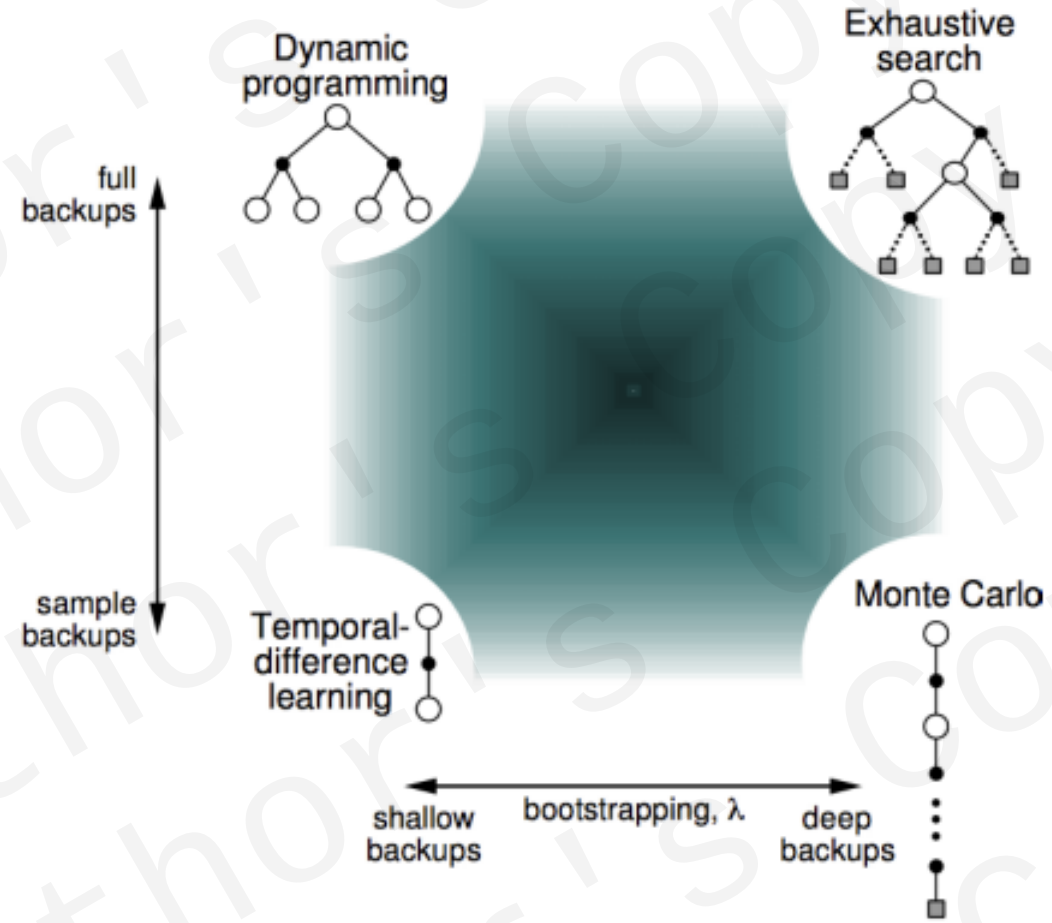
- TD can learn before final outcome at every step.
- MC must update after completion.
- TD can learn
  - without final outcome
  - from incomplete sequences
  - in continuing environments.
- MC only works for episodic environments.
- TD exploits Markov property (efficient in Markov Environments)



Dynamic Programming Back up.

$$V(S_t) \leftarrow \mathbb{E}_{\pi}\left(R_{t+1} + \gamma V(S_{t+1})\right)$$





Model-free Control

TD control (On-policy SARSA)

TD control (Off-Policy) Q-learning

JHA Mayank , Email: mayank-shekhar.jha [at] univ-lorraine.fr

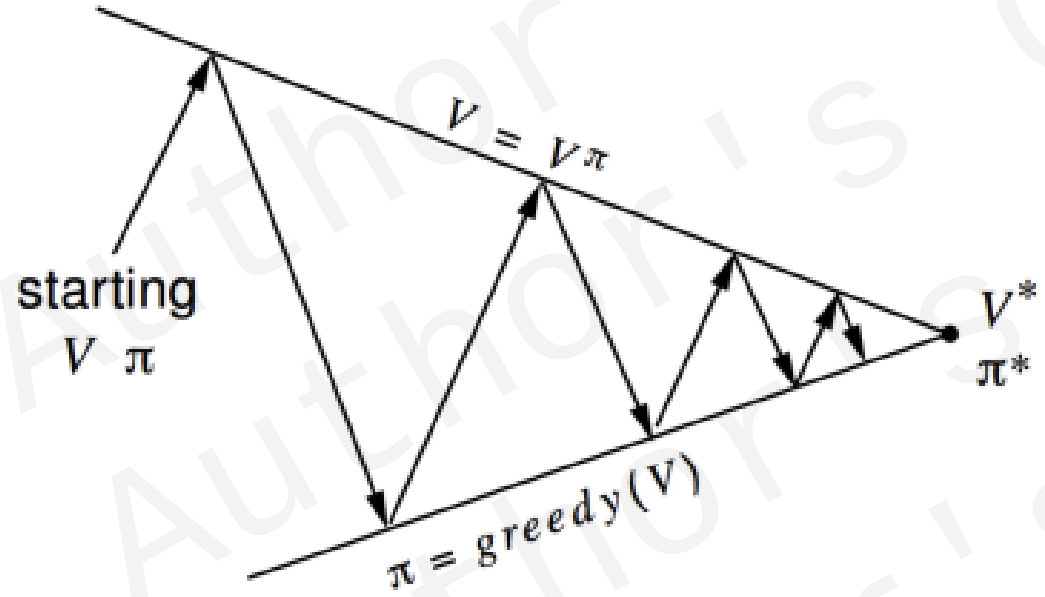
Model Free Control

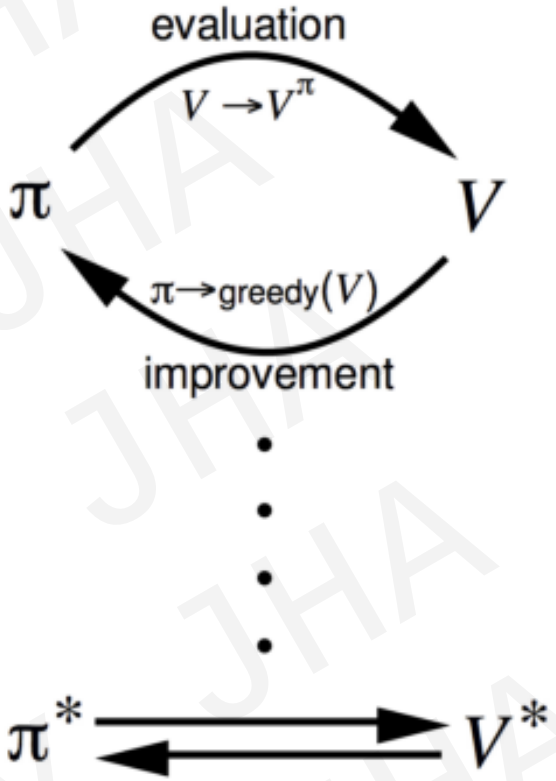
→ MDP model is unknown,
experience sampling is possible!

On-Policy learning : learn while on 'RUN'
learn about policy π, from experience sampled from π

Off-Policy learning : learn while off-line.
learn about policy π from experience sampled from μ.

Generalised Policy Iteration





Policy Evaluation : Evaluate V_π (Ex: Iterative Policy Evaluation)

Policy Improvement: Generate $\pi' \geq \pi$ (Ex: Greedy Policy Improvement)

What is Greedy strategy ??

Greedy Policy Improvement Example:

$$\pi'(s) = \arg\max_{a \in A} Q(s,a)$$

choose the action that maximises the Q-fun value.

the action chosen is greedy with respect to our objective

Chooses action that gives max

maximise the value!

What is limitation of greedy policy → can get stuck at local minimas!

Imagine being greedy but not exploring all state space!

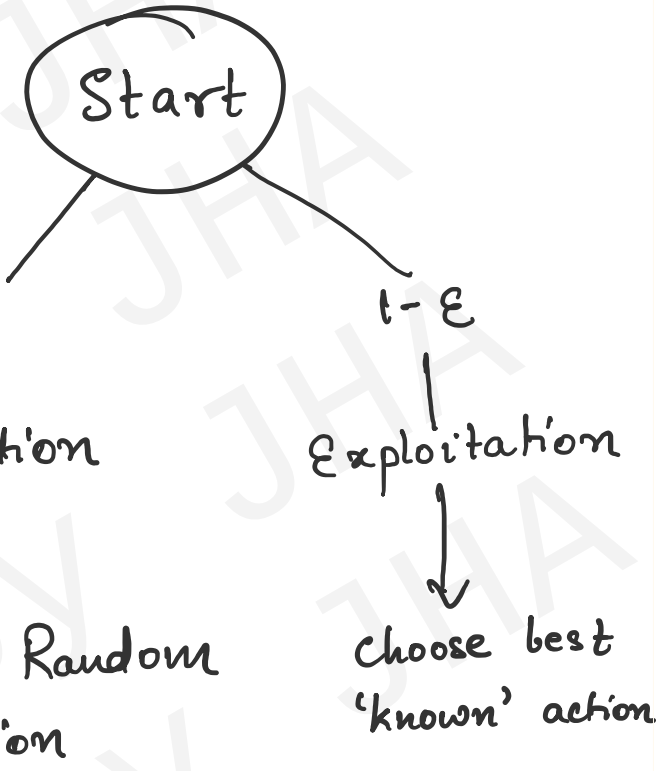
Exploration

ε- Greedy Exploration.

ε-Greedy satisfies the following conditions

→ Choose greedy action : with probability $(1-\varepsilon)$

→ choose random action : with probability ε.



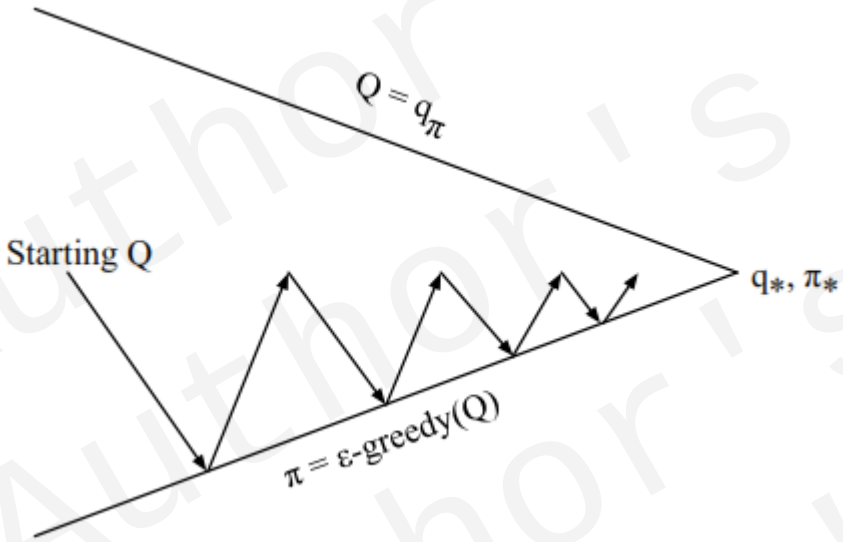
Greedy in Limit with Infinite Exploration (GLIE)

- → all state-action pairs are explored infinitely many times.
- → Policy converges on a greedy policy in asymptotic sens

$$\lim_{k \to \infty} \pi_k(a|s) = 1 \left(a = \underset{a' \in A}{\operatorname{argmax}} \, Q_k(s,a') \right)$$

Note: ε-greedy with $\varepsilon_k = 1/k$ is GLIE.

Policy Iteration (TD control)



Idea:

1) Apply TD to update $Q(s,a)$

2) Use ϵ-greedy policy improvement

3) Update every time step!

Every time step:

- Policy Evaluation: Use TD to update $Q(s,a)$

$$Q(S,a) \leftarrow Q(S,a) + \alpha\left(R + \gamma Q(S',A') - Q(S,A)\right)$$

- Policy Improvement: ϵ-greedy policy improvement

TD Control (Update Eqn)

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

$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$ → TD target

$Q(S_t, A_t)$ → current estimate

TD Control (Algorithm)

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

Initialize : $Q(s,a)\ \forall s \in S, a \in A$. → initializ with all action-values to zero.

$\pi \leftarrow \epsilon\text{-greedy}(Q)$ → choose an initial policy

$S_0\ A_0\ R_1$

→ interact with environment, take action A_0

→ obtain the reward.

TD Control (Algorithm)

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

Initialize : $Q(s,a)\ \forall s \in S, a \in A.$ → initializ with all action-values to zero.

$\pi \leftarrow \epsilon\text{-greedy}(Q)$ → choose an initial policy

$S_0\ A_0\ R_1\ S_1\ A_1$

→ next state is generated

→ Use same π to generate next action A_1

TD Control (Algorithm)

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

$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$: TD target

$Q(S_t, A_t)$: current estimate

Initialize : $Q(s,a) \; \forall s \in S, a \in A$. → initialize with all action-values to zero.

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

$S_0 \; A_0 \; R_1 \; S_1 \; A_1 \; | \; R_2 \; S_2 \; A_2 \; \ldots\ldots \; S_t \; A_t \; R_{t+1} \; S_{t+1} \; A_{t+1}$

Update $Q(S_0, A_0)$: $Q(S_0, A_0) \leftarrow Q(S_0, A_0) + \alpha \left(R_1 + \gamma Q(S_1, A_1) - Q(S_0, A_0) \right)$

(update the value — policy-evaluation)

$\pi \leftarrow \epsilon\text{-greedy}(Q)$ → choose $a = \underset{a \in A}{\text{argmax}} \; Q(S_0, A_0)$ → policy improvement.

TD Control (Algorithm)

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

$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$: TD target

$Q(S_t, A_t)$: current estimate

Initialize : $Q(s,a) \; \forall s \in S, a \in A.$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

$S_0 \; A_0 \; R_1 \; S_1 \; A_1 \; | \; R_2 \; S_2 \; A_2 \; | \; \ldots\ldots \; S_t \; A_t \; R_{t+1} \; S_{t+1} \; A_{t+1}$

Update $Q(S_0, A_0)$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

Update $Q(S_1, A_1)$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

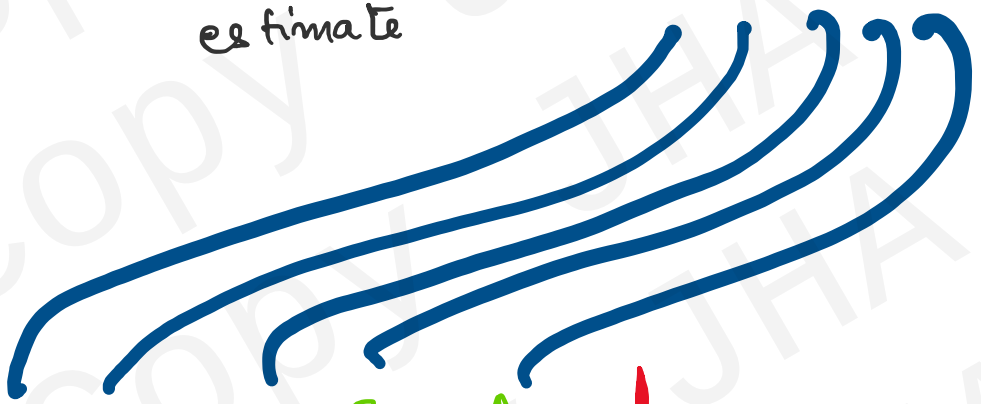
TD Control (Algorithm) SARSA (on-policy)

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

$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$: TD target

$Q(S_t, A_t)$: current estimate

SARSA



Initialize : $Q(s,a)\ \forall s \in S, a \in A.$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

$S_0\ A_0\ R_1\ S_1\ A_1 \mid R_2\ S_2\ A_2 \mid \ldots\ldots\ S_t\ A_t\ R_{t+1}\ S_{t+1}\ A_{t+1}$

Update $Q(S_0, A_0)$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

Update $Q(S_1, A_1)$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

Update $Q(S_t, A_t)$

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

Off-policy Learning

Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$

while following another policy $\mu(a|s)$

Advantages:

1. Learn about optimal policy while following exploratory policy.
2. Learn about multiple policies while following one policy.
3. Re-use experience generated from old policies $\pi_1, \pi_2, \pi_3, \ldots$
4. Learn by observing other agents.

TD Control (Off-policy) Q-Learning

→ Off-policy learning of $Q(s,a)$ [Q-function]

→ Next action is chosen using a behaviour policy $A_{t+1} \sim \mu(\cdot|S_t)$

→ But, best successor action chosen $a' \sim \pi(\cdot|S_t)$

→ Update $Q(S_t,A_t)$ towards value given by alternative action!

$$Q(S_t,A_t) \longleftarrow Q(S_t,A_t) + \alpha\left(R_{t+1} + \gamma Q(S_{t+1},a') - Q(S_t,A_t)\right)$$

best alternative action

Q-learning (Algorithm).

→ Consider two policies → Behaviour → used for exploration → improve!!
→ Target → that is learnt

→ Behaviour policy : μ : ϵ-greedy wrt $Q(s,a)$ ← explores, exploits

→ Target policy : π : greedy wrt $Q(s,a)$ ← always selects 'max'

→ Q-learning target becomes:

$$R_{t+1} + \gamma Q(S_{t+1}, A')$$

$$= R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a' \in A}{\operatorname{argmax}}\, Q(S_{t+1}, a')\right)$$

$$= R_{t+1} + \max_{a' \in A} \gamma Q(S_{t+1}, a')$$

$$\pi(S_{t+1}) = \arg\max_{a} Q(S_{t+1}, a)$$

SARSA (on-policy) Reminder

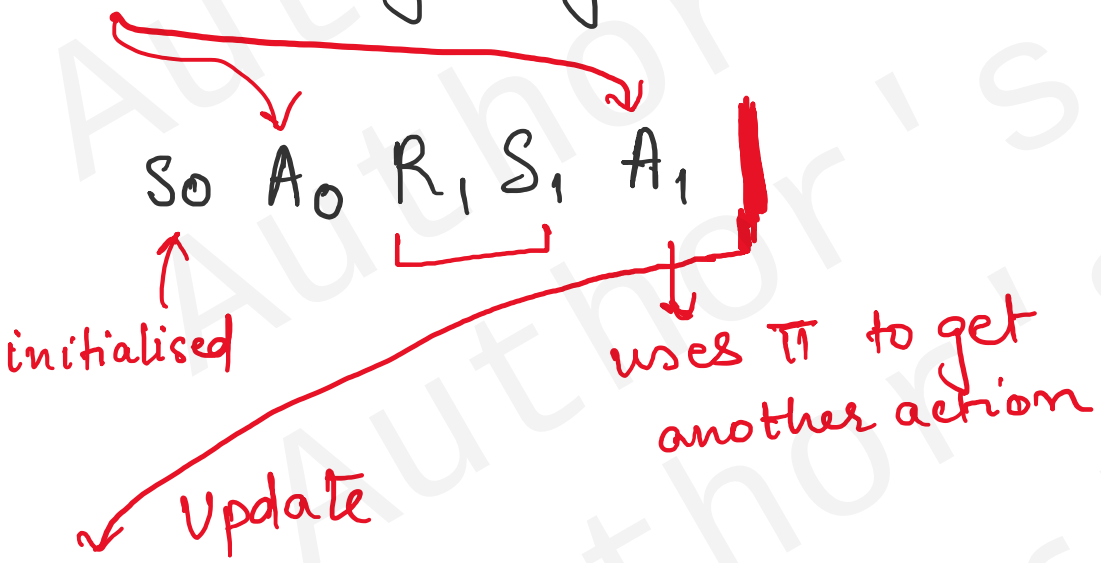
Initialize $Q(s,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)

SARSA (on-policy) Reminder

Initialize $Q(s,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)



$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha\left(R_1 + \gamma Q(S_1,A_1) - Q(S_0,A_0)\right)$$

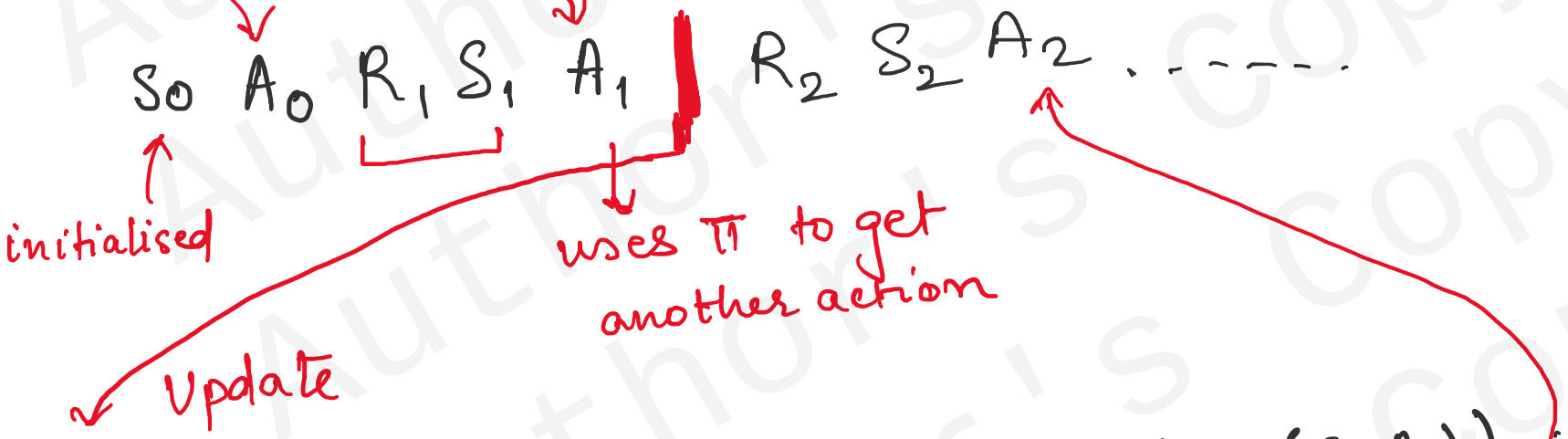
and improve to most recent update

$\pi \leftarrow \epsilon$-greedy(Q)

SARSA (on-policy) Reminder

Initialize $Q(s,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)



$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha\left(R_1 + \gamma Q(S_1,A_1) - Q(S_0,A_0)\right)$$

and improve to most recent update

$\pi \leftarrow \epsilon$-greedy(Q)

Q-learning = SARSA (max)

Initialize $Q(S,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)

$S_0\ A_0\ R_1\ S_1$

initialised — Update

$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha\left(R_1 + \gamma Q(S_1, \_) - Q(S_0,A_0)\right)$$

① What action to use here?

② SARSA uses ϵ-greedy

③ Here, we could maximise direct

keep $\pi \sim \arg\max_{a \in A} (Q(s,a')$

ie. use greedy policy.

Q-learning = SARSA (max)

Initialize $Q(s,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)

$S_0\ A_0\ R_1\ S_1$

initialised

Update

$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha\left(R_1 + \gamma Q\left(S_1, \underset{\grave{a} \in A}{\arg\max}\, Q(S_1,\grave{a})\right) - Q(S_0,A_0)\right)$$

keep $\pi \sim \underset{\grave{a} \in A}{\arg\max}\, (Q(s,a')$

ie. use greedy policy.

Q-learning = SARSA (max)

Initialize $Q(S,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)

S_0 A_0 R_1 S_1

initialised

Update

$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha\left(R_1 + \gamma \max_{a \in A} Q(S_1,a)\right) - Q(S_0,A_0)$$

keep $\pi \sim \arg\max_{a \in A} (Q(S,a')$

ie. use greedy policy.

Q-learning = SARSA (max)

Initialize $Q(S,a) = 0$ for all $s \in S$, $a \in A$

$\pi \leftarrow \epsilon$-greedy(Q)

$S_0\ A_0\ R_1\ S_1 \mid A_1\ R_1\ S_1$

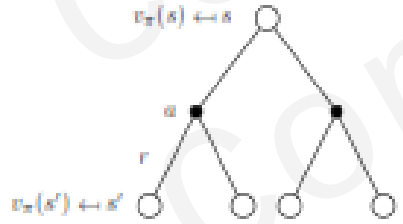
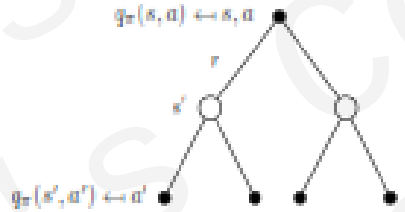
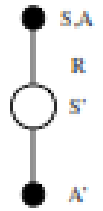
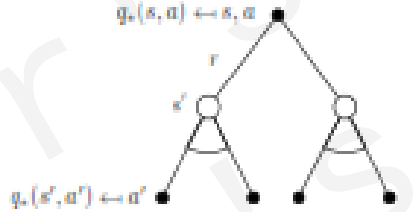
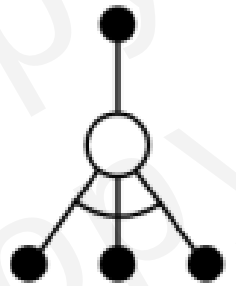
initialised ↑ Update ↓

$$Q(S_0,A_0) \leftarrow Q(S_0,A_0) + \alpha\left(R_1 + \gamma \max_{a \in A} Q(S_1,a)\right) - Q(S_0,A_0)$$

$\pi \leftarrow \epsilon$-greedy (updated Q)

Q-learning → attempts to directly estimate the optimale value at every time step !!

(max)

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s,a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s,a)$	 Q-Value Iteration	 Q-Learning

Summary

Full Back up DP

Iterative Policy Evaluation

$$V(s) \leftarrow \mathbb{E}\left(R + \gamma V(s') \mid s\right)$$

Q-Policy Iteration

$$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(s',A') \mid s,a\right]$$

Q-value Iteration

$$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in A} Q(s',a') \mid s,a\right]$$

Sample Back-Ups

TD learning

$$V(s) \leftarrow V(s) + \alpha\left(R + \gamma V(s')\right)$$

SARSA

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$$

Q-learning

$$Q(S,A) \leftarrow Q(S,A) +$$

$$\alpha$$