Model-Free Control

Reinforcement Learning · Lecture 4

Christopher Mutschler

University of Technology Nuremberg

5 November 2026

Today

1. **Recap:** MC, TD and backup diagrams
2. **From prediction to control:** value functions and policy improvement
3. **Exploration:** ε-greedy action selection
4. **SARSA and Q-Learning:** on- and off-policy TD
5. **Cliff Walking:** what changes when exploration is risky?
6. **Extensions & takeaways:** Double Q-Learning, convergence, and scaling up

TODAY'S QUESTION

We can estimate $Q^\pi(s, a)$ from experience. Can we **learn a better policy** while collecting that experience?

PART 1

Recap: prediction from data

Monte Carlo, temporal differences, and the anatomy of a backup.

RECAP

Last time: the policy was fixed

LECTURE 3 · MODEL-FREE PREDICTION

Follow a given π and estimate how good it is:

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

MC averages complete returns. TD bootstraps after one step.

LECTURE 4 · MODEL-FREE CONTROL

The policy is **not fixed** anymore.

Estimate Q from experience and improve π along the way.

A KEY TRANSITION

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\,\delta_t$$

The target δ_t now depends on **which policy we want to learn**.

RECAP

MC versus TD(0): what changes?

MONTE CARLO PREDICTION

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

Target: complete observed return G_t

After the **episode** · no bootstrap · larger target variance

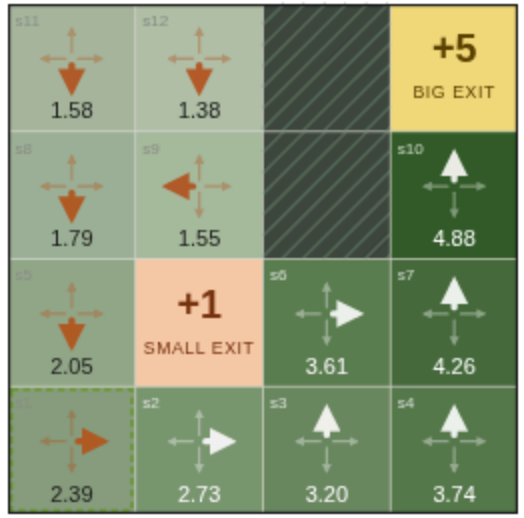
TEMPORAL-DIFFERENCE PREDICTION

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

Target: next reward + estimated value of the next state

Every **time step** · bootstrap · can be biased

THE EXACT TWO-EXIT MAZE · LECTURE 3



SAME FIXED POLICY, SAME EXPERIENCE

Start $\mathbf{s_1}$. Two possible final rewards: **+1** (small exit) and **+5** (big exit).

Example episode: $\mathbf{s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_7 \rightarrow s_{10} \rightarrow +5}$.

MC propagates the **full sampled return** along the episode. TD(0) updates **one transition at a time** using its estimate of the next state.

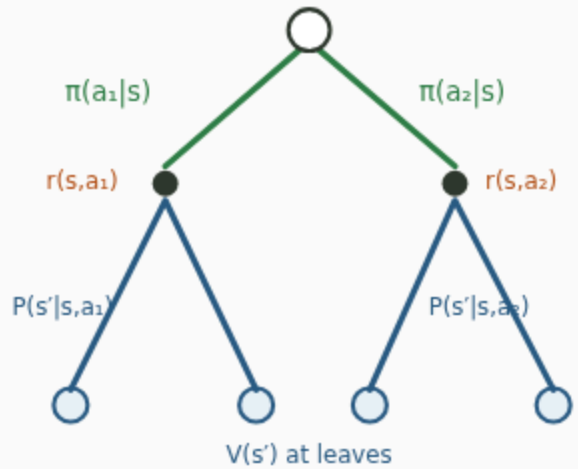
$\gamma = 0.9$, deterministic moves, rewards only on exit

RECAP

Backup diagrams: what each target looks at

Dynamic Programming

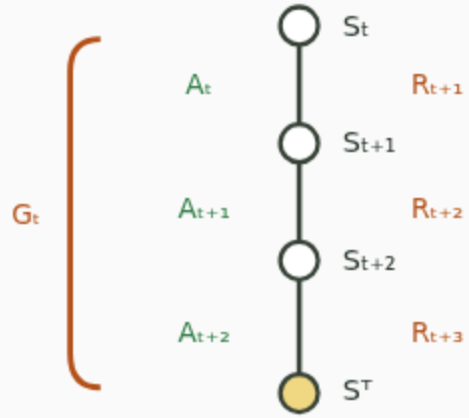
$$\sum_a \pi(a \mid s)\Big[r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s')\Big]$$



Model-based **expectation over all actions and successor states**.

Monte Carlo

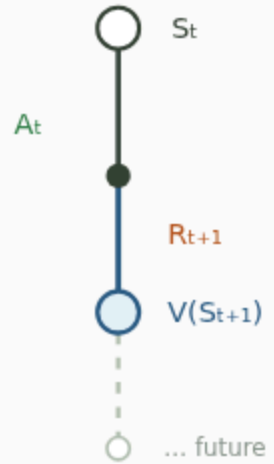
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$



One observed trajectory through episode termination.

TD(0)

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



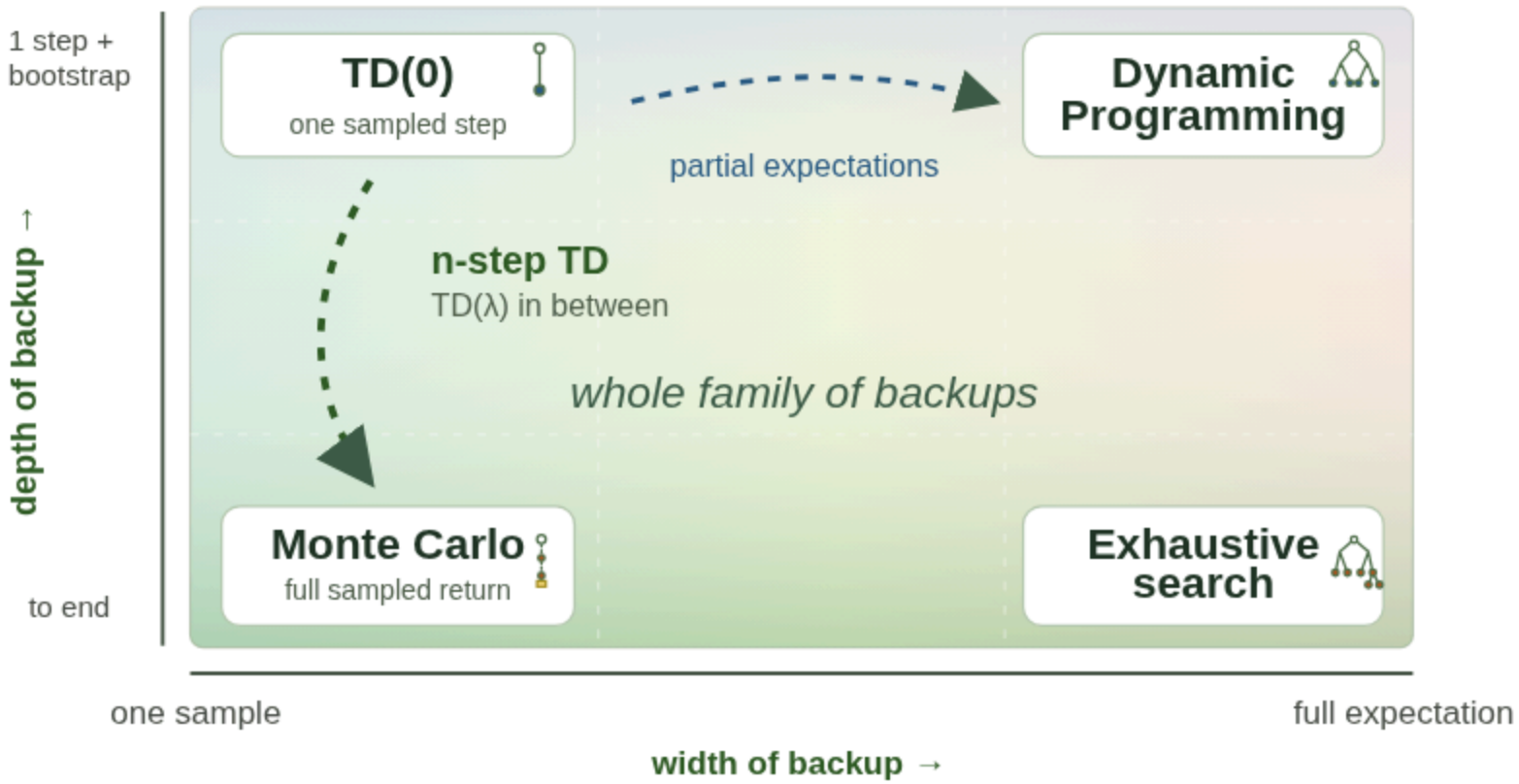
One observed transition, then bootstrap from a stored value.

● Actions selected by π ● Rewards received ● Values and model transitions

RECAP

A unified view of prediction algorithms

Vary **backup depth** continuously (how many steps before bootstrapping) and **backup width** (how many successor branches are considered).



SAMPLING ↔ EXPECTATION

From **one observed successor** to the **full model-based average**.

BOOTSTRAP ↔ FULL RETURN

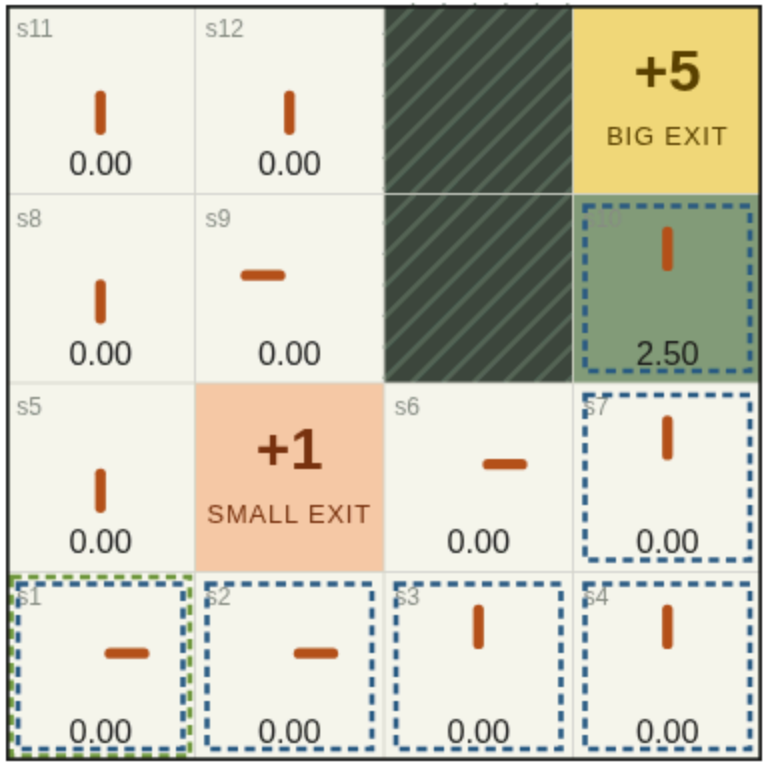
From **one-step TD** to longer **n-step** returns and **Monte Carlo**.

The axes are continua, not four unrelated algorithms. Adapted from Sutton & Barto (2018), Fig. 6.3.

RECAP · SAME MAZE

TD(0): update after each step

Follow the **same 6-transition episode** from Lecture 3. Start with $V_0 = 0$, learning rate $\alpha = 0.5$, discount $\gamma = 0.9$.



Step → one transition | Finish episode | Reset

TEMPORAL-DIFFERENCE UPDATE

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

LIVE UPDATE

Step 6: s_{10} ↑ terminal exit (+5)

$V(s_{10}) \leftarrow 0.000 + 0.5\,[5.000 + 0.9 \times 0.000 - 0.000]$

= 2.500

TD target: 5.000 | TD error: 5.000

Episode complete — the next step starts a new episode.

Episode 1 · transition 6 / 6

WHAT SHOULD YOU OBSERVE?

After the **first** episode, only **$V(s_{10})$** has changed: **2.50**. Information moves backwards through later episodes.

PART 2

From prediction to control

Generalised policy iteration, action values, and exploration.

Prediction vs. control

Predict the future given a policy π.

(estimate the value function)

PREDICTION · FROM LAST WEEK

Input: a fixed policy π
Output: state values V^π
Question: how good is this policy?

Evaluation: **estimate** expected returns from sampled trajectories.

Optimize the future by finding the best policy π.

(optimize the value function)

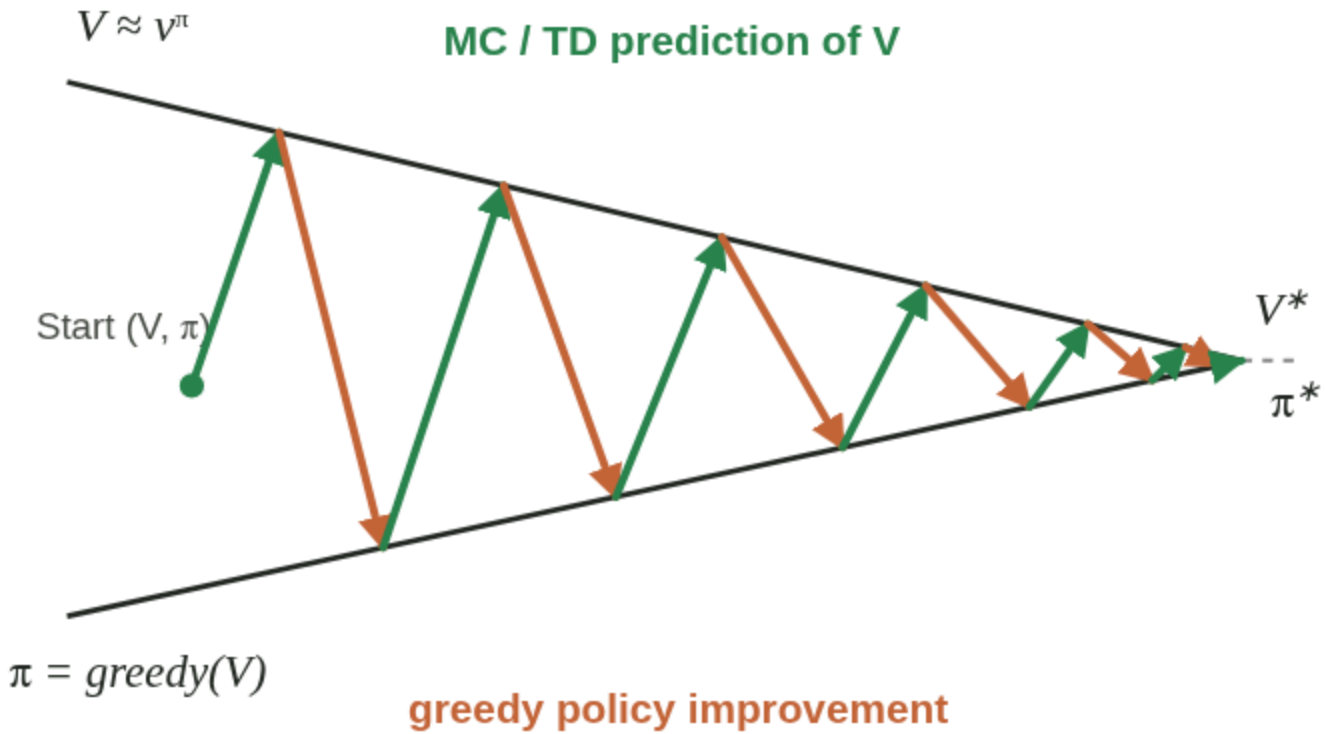
CONTROL · WHAT WE WANT NEXT

Input: experience *(s,a,r,s')*
Output: an improved policy π'
Question: can we improve actions using V^π?

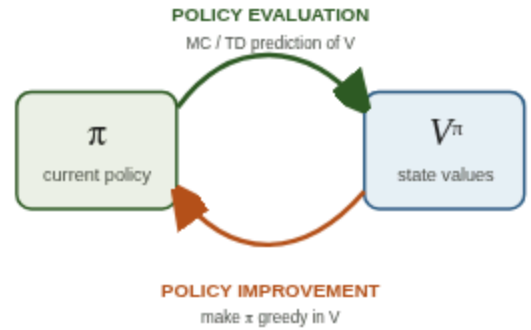
Evaluation and improvement **alternate**, often every time step.

The natural idea: GPI with V^π

Start with an arbitrary policy and state-value estimate. Alternate **MC/TD prediction** with **greedy policy improvement**.



GENERALIZED POLICY ITERATION



↗ **MC / TD prediction** of V^π

↘ **Greedy policy improvement**

Repeated evaluation and improvement moves us toward V^* and π^*.

The catch: greedy improvement over V

With a known model, we can improve a policy using V^π:

$$\pi'(s) \in \arg\max_a \left[r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^\pi(s') \right]$$

To choose an action, we need the one-step dynamics P and r — precisely what we do not know!

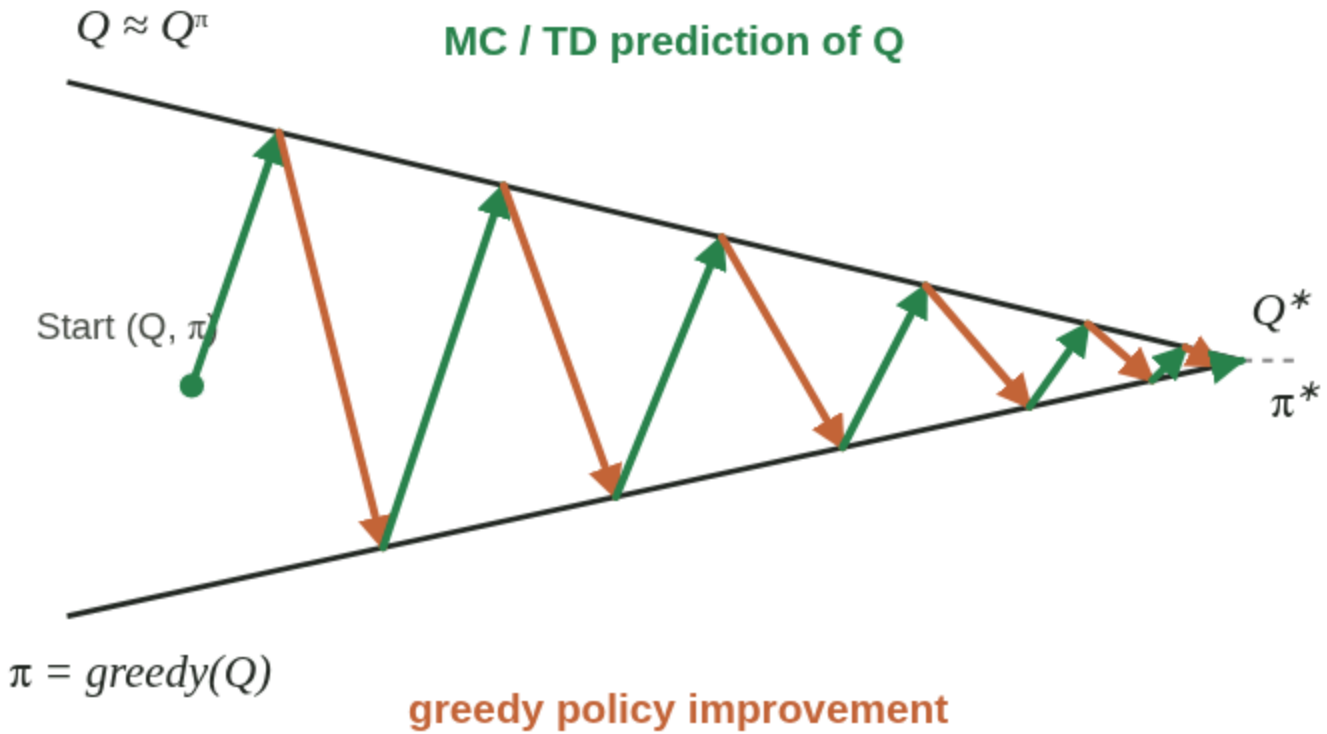
HOW CAN WE AVOID NEEDING A MODEL?

What if our value estimate already included the **first action**?

REMEMBER LECTURE 3?

Action values Q^π

The first action is already included in Q^π. So we can replace the model-dependent improvement step by $\pi'(s) = \arg\max_a Q^\pi(s, a)$.



FROM LAST LECTURE

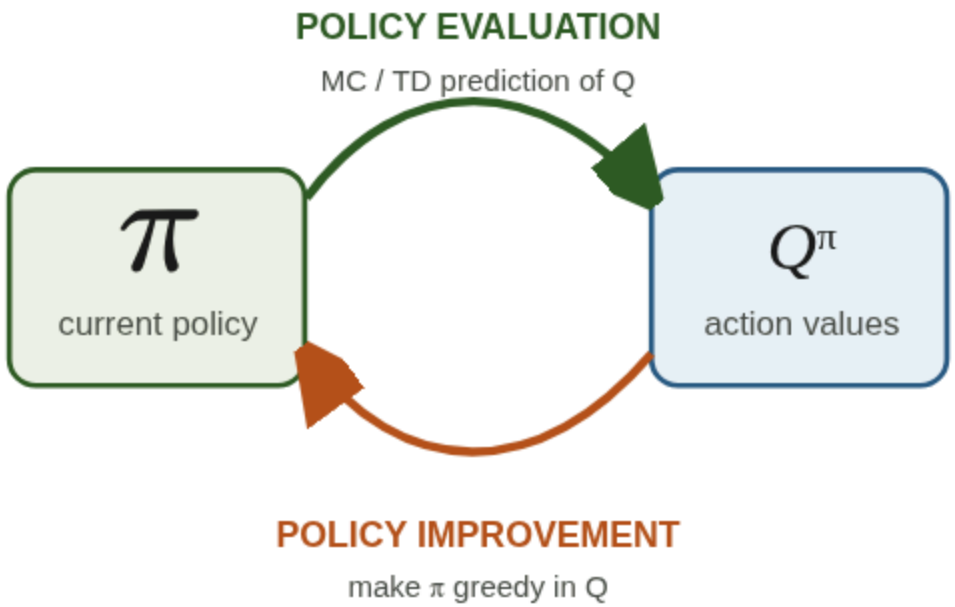
Action values for a **fixed policy**:
$Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$
TD can estimate them from samples.

THE MODEL-FREE SHORTCUT

Improve by choosing the action with the highest learned Q^π — without knowing P or R.

Same GPI arrows as before, now predicting Q rather than V.

Generalised policy iteration



- **Policy evaluation:** estimate Q^π from sampled transitions.
- **Policy improvement:** adjust π using the new action values.
- Unlike full DP, both happen **incrementally**, without knowing P or R.

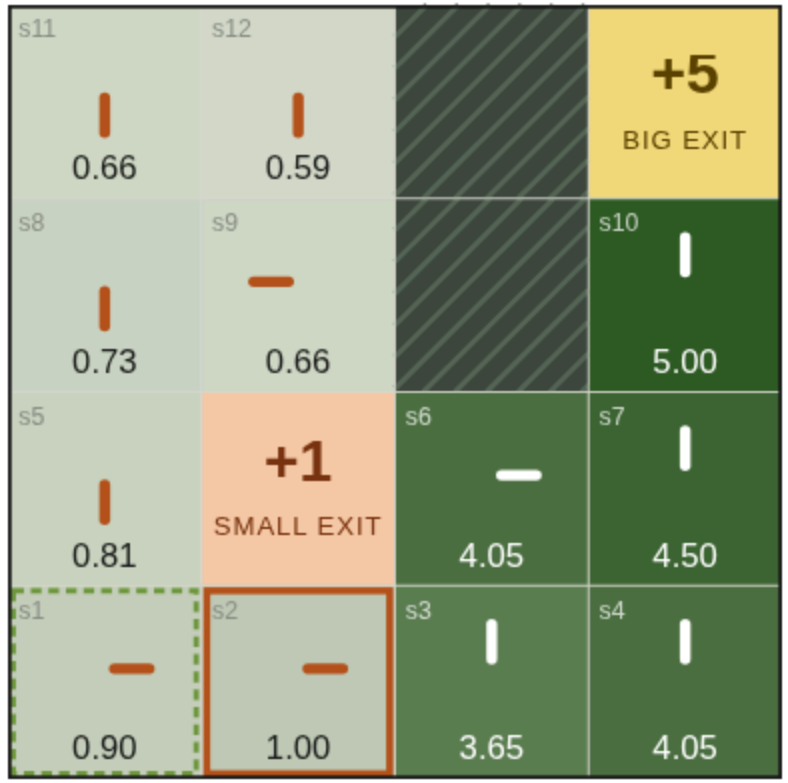
KEY IDEA

A control algorithm is essentially **prediction + an improving policy**.

Greedy policy improvement: revisit the maze

In the same two-exit maze, we start with a **fixed policy and its already evaluated values**. Click a cell to improve only its action; evaluation is a separate step.

TWO-EXIT MAZE · EVALUATED POLICY



The displayed values come from the last policy evaluation. $\gamma = 0.9$.

1 · SELECT A STATE

S_2 Current π(s): → · best from stored Q: →

↑ up		**1.00**
→ **right**		**3.28**
↓ down		**0.90**
← left		**0.81**

STEP 1 · POLICY IMPROVED · Only $\pi(s_2)$ changed. **V and Q are still frozen.**

Improve only this state | **Evaluate updated policy** | **Reset π**

Improvement at s_2: π(s) ↑ → →. All other policy arrows stay as before. **V and Q have not changed.** Click "Evaluate updated policy" to update the values.

NOW THE KEY QUESTION

What if a better action was never tried?

The exploration trap: lucky first samples

Two actions have **unknown Gaussian rewards**. Try each once, then keep choosing the action with the **higher sample average**.

ONE STATE · TWO ACTIONS

ACTION A
sample reward

ACTION B
sample reward

Action A · 0 samples Q = —

Action B · 0 samples Q = —

Sample each once | **Continue greedy (+40)**

Reset

Start with one reward sample from each action.

REWARD SAMPLES FROM NORMAL DISTRIBUTIONS

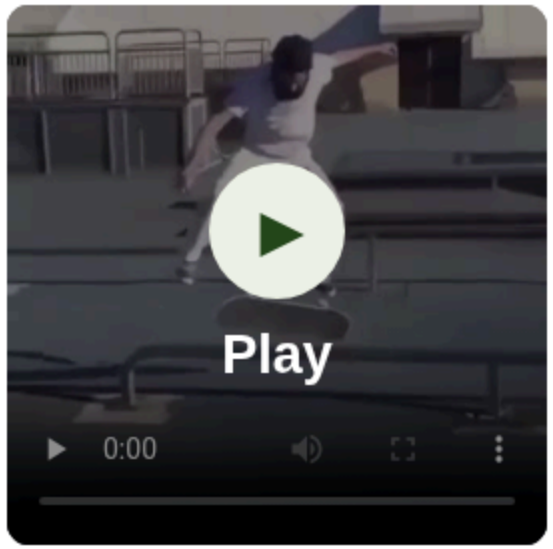


No samples yet.

Reveal true reward distributions

B has the larger true mean, but unlucky first samples can hide it.
Grey marks = sampled rewards.

EXPLORATION IN REAL LIFE :)



Sometimes you only discover a better move by **trying something new**.

Epsilon-greedy exploration

With probability $1-\varepsilon$, choose a greedy action. With probability ε, choose **uniformly at random**.

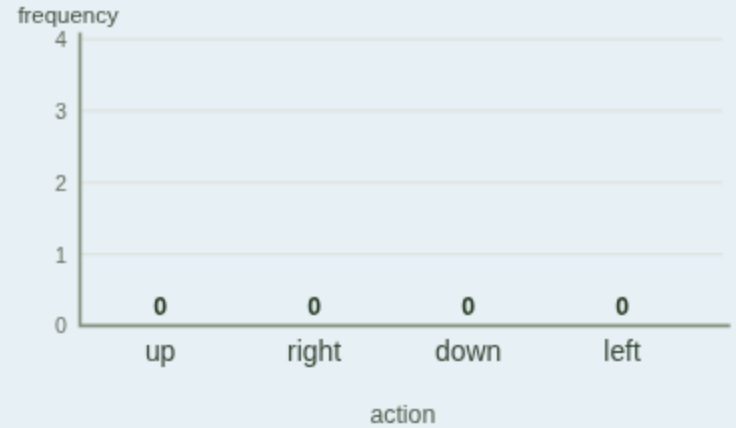
In this example, ↑ up is the greedy action (it currently has the highest estimated Q-value).

$$\pi_\varepsilon(a \mid s) = \frac{\varepsilon}{|\mathcal{A}|} + (1-\varepsilon)\,\mathbf{1}[a = a^*]$$

For a unique maximiser a^*; with ties, split the greedy mass across maximisers.

Exploration rate ε 0.20

ACTION FREQUENCY · OBSERVED SAMPLES



Sample once | +100 samples | Reset

0 sampled actions. Expected probability of up: 85.0%.

The histogram counts actions actually chosen; it approaches the ε-greedy probabilities with more samples.

GLIE: ε decreases, but exploration continues

Same four actions as on the previous slide. Greedy action: **up**. Set the exploration-rate scale ε_0. Then $\varepsilon_t = \min(1, \varepsilon_0 / \sqrt{t})$ gradually decreases.

GREEDY IN THE LIMIT WITH INFINITE EXPLORATION

1. Each action is sampled infinitely often: $N_t(s, a) \to \infty$.

2. Exploration probability tends to zero: $\varepsilon_t \to 0$.

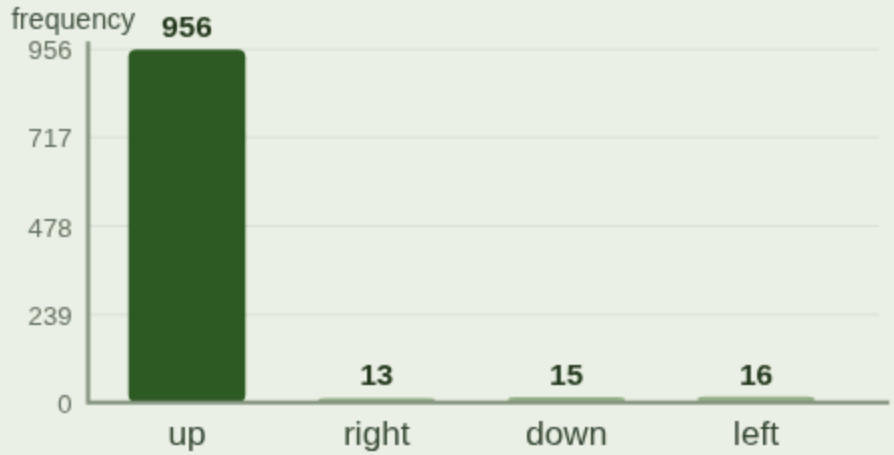
For every $\varepsilon_0 > 0$, the scale $\varepsilon_0/\sqrt{t}$ tends to zero but the sum of the exploration probabilities diverges (for this single-state example).

Exploration-rate scale ε_0 1.0

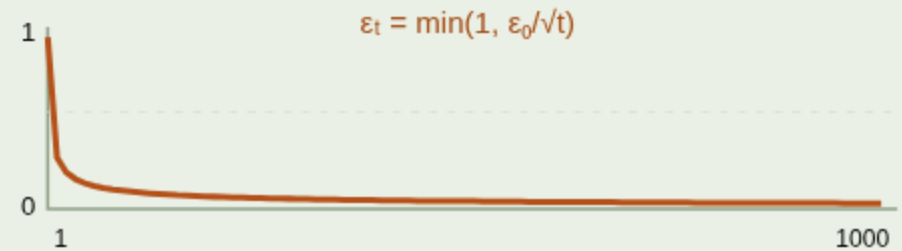
+1 step | +100 steps | +1,000 steps | Import previous counts | Reset

t = 1000 · ε_t = 0.032 · up frequency = 95.6% · other actions sampled 44 times.

OBSERVED ACTION FREQUENCIES



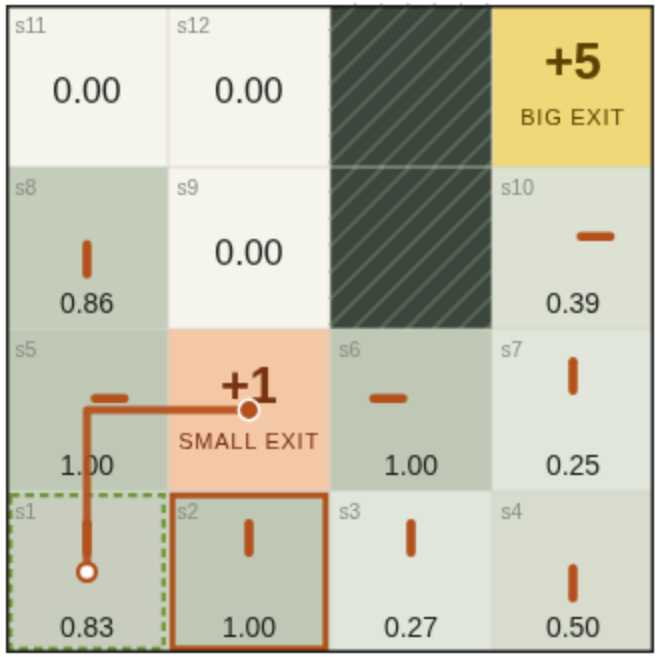
EXPLORATION RATE OVER TIME



Monte Carlo control: improve after full episodes

Try **full episodes** in the **Maze environment**. Each episode updates sampled returns *G* for state–action pairs and improves the **ε-greedy policy**.

MAZE ENVIRONMENT · 4 × 4



Click a numbered cell to inspect learned Q-values. Orange path: the last sampled episode. Arrows: the current greedy direction.

Exploration-rate scale ε_0 2.0

+1 episode **+30** **+300** **Reset**

MC CONTROL LAB

S_2 30 episodes

↑ up	**1.00**
→ right	**0.16**
↓ down	**0.57**
← left	**0.50**

Last: start s_1, 5 moves, exit reward +1, first return 0.66.
Exploration rate ε = 0.37
Episodes reaching an exit: 30 / 30.

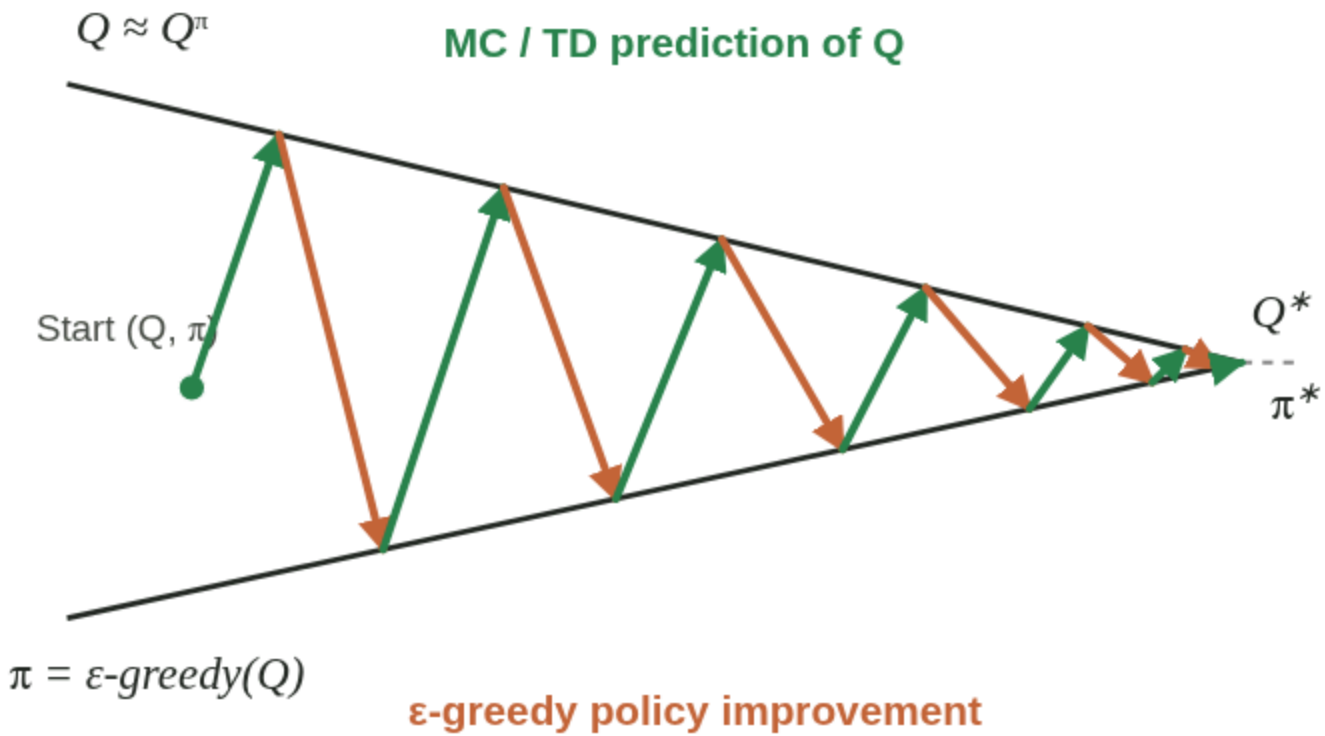
1. Generate episode → 2. Compute returns → 3. Update Q → 4. Make π ε-greedy

GPI cycle: 30 Monte Carlo evaluations → 30 ε-greedy improvements

Every episode begins at s_1; $\varepsilon_k = \min(1, \varepsilon_0 / \sqrt{k})$, $\gamma = 0.9$. Model-free tabular first-visit MC control.

GPI now uses ε-greedy policy improvement

Purely greedy control can miss better actions. **ε-greedy improvement** keeps exploring while we learn a better policy.



WHAT CHANGES?

Evaluation still estimates Q^{π} from MC/TD experience.
The improved π continues to explore.

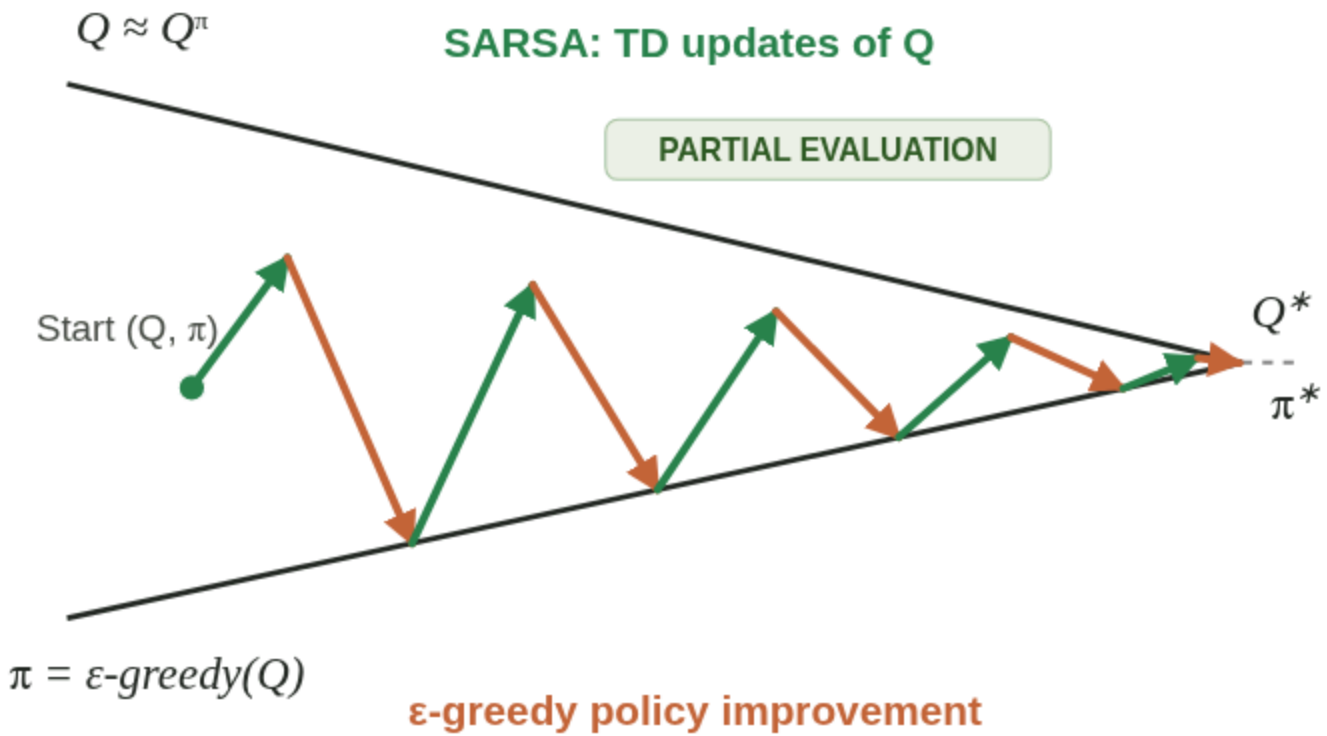
FROM OUR MAZE LAB

0 completed episodes

Every full episode supplies returns, then Q and the ε-greedy policy are updated.

We do not have to finish policy evaluation

Every episode: MC can improve after one sampled return. **Every step:** TD control can improve even faster.



GENERALISED POLICY ITERATION

Each short green arrow is just a **partial evaluation**; then orange improves the policy. We alternate before reaching Q^π exactly.

SPOILER: SARSA

TD prediction of Q + **ε-greedy policy improvement after every time step**.

EVERY EPISODE → EVERY STEP

PART 3

SARSA

On-policy temporal-difference control: learn from the action you actually take.

From TD prediction to TD control

Last lecture: one-step TD for action values, **under a fixed policy**:

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

ONLY TWO CHANGES

1. Select actions with an **ε-greedy policy** derived from the current Q.
2. Update Q and therefore the policy continuously.

Observe → Update **Q** → Act **ε-greedily**

No model and no full-episode return needed.

SARSA: the update

$$\boxed{Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\, \delta_t}$$

$$\delta_t = \underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})}_{\text{TD target}} - Q(S_t, A_t)$$

CURRENT PAIR

State
Action

OBSERVED

Reward

NEXT PAIR

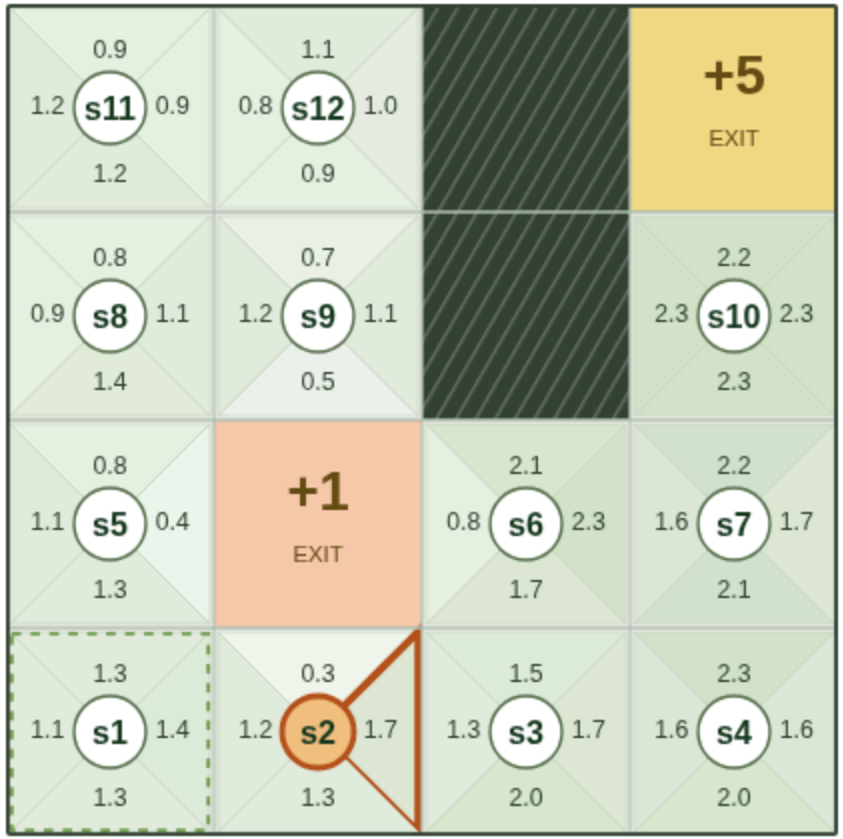
State
Action

S-A-R-S-A = state, action, reward, next state, **next sampled action**.

Let us calculate one SARSA update in the maze

In the **same Maze environment**, each state contains four action values. **Click a state or directional triangle**, observe the transition, then update one Q-value.

4×4 MAZE · 4 ACTION VALUES IN EACH CELL



The highlighted triangle is the selected action.

INTERACTIVE SARSA BACKUP

S_2 · → $\alpha = 0.5, \gamma = 0.9$

$S_2 \rightarrow \rightarrow S_3$

Next action from π: ↓

Previous **Q(s,a)**	**1.677**
Observed reward **R**	**0**
Sampled π action Q^{π}	**1.959 [↓]**
TD target	**1.763**
Updated Q(s,a)	**1.720**

Apply SARSA update **Sample next action** **Reset Q**

0 updates made. Choose a cell and one of its four action-value triangles.

SARSA in six lines

ON-POLICY TD CONTROL

```
Initialize Q(s,a) arbitrarily; choose α, γ, ε
For each episode:
  Initialize S; choose A ~ ε-greedy(Q(S,·))
  Repeat:
    Take A, observe R and S'
    If S' is not terminal, choose A' ~ ε-greedy(Q(S',·))
    Q(S,A) ← Q(S,A) + α [R + γQ(S',A') − Q(S,A)]
    S ← S',   A ← A'
  Until S is terminal
```

Interpret Q(terminal,·) = 0.

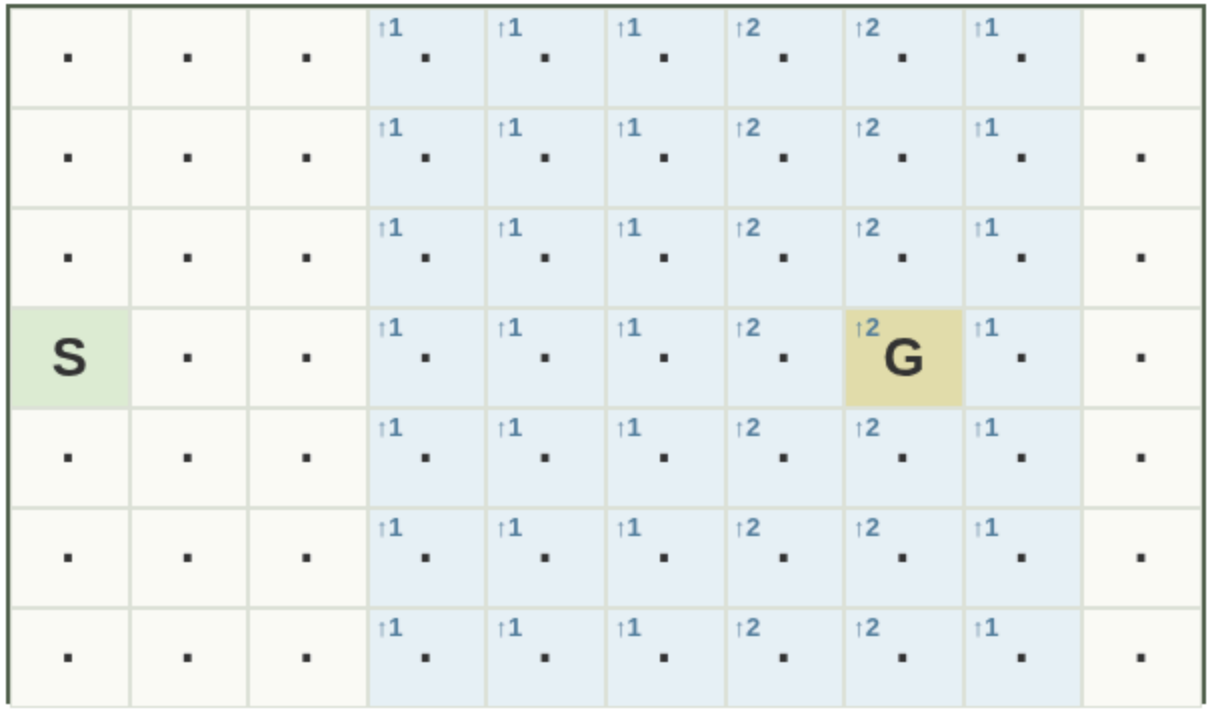
SPOT THE CRUCIAL LINE

Why do we have to choose **A'** before updating Q?

Windy Gridworld: learn with SARSA

7 × 10 grid · Start S → Goal G · updrafts in columns 4–9

reward = −1 / step · $\gamma = 1$



↑ blue: wind strength black arrows: current greedy policy

orange: actual trajectory of the last completed episode

+1 complete episode | +20 | +200 | Reset

0 completed episodes · not trained



Every counted episode **ends at G**. The orange lines show the actual experienced trajectory, not the current greedy policy. How does SARSA compensate for the wind without knowing its dynamics?

Why SARSA is *on-policy*

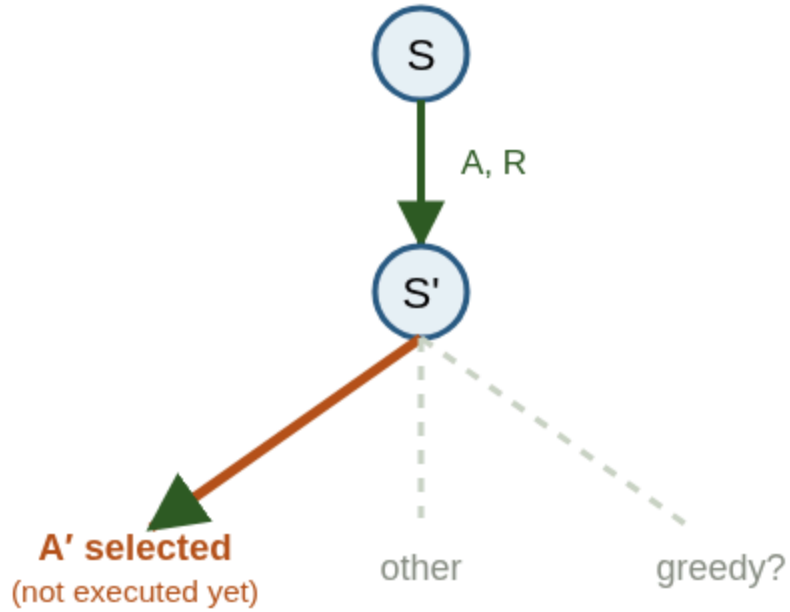
The target contains the value of the action **actually chosen** by the same policy:

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

With $\varepsilon > 0$, A_{t+1} can be **exploratory**, even when there is a better greedy action.

INTERPRETATION

SARSA learns the action values of its **current exploratory behaviour**. It accounts for future mistakes caused by exploration.



SARSA: strengths and trade-offs

STRENGTHS

- Processes each transition **immediately**
- Cheap tabular update
- Directly incorporates the risks of its current policy

LIMITATIONS

- Action samples must come from the **current** behaviour policy
- Needs sustained exploration and many samples
- Sensitive to learning-rate and exploration schedules
- Can be unstable with naive function approximation

PART 4

Q-Learning

Off-policy TD control: learn about the greedy policy while exploring.

Q-Learning: change one term

SARSA $$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

Q-Learning $$R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$

THE COMPLETE UPDATE

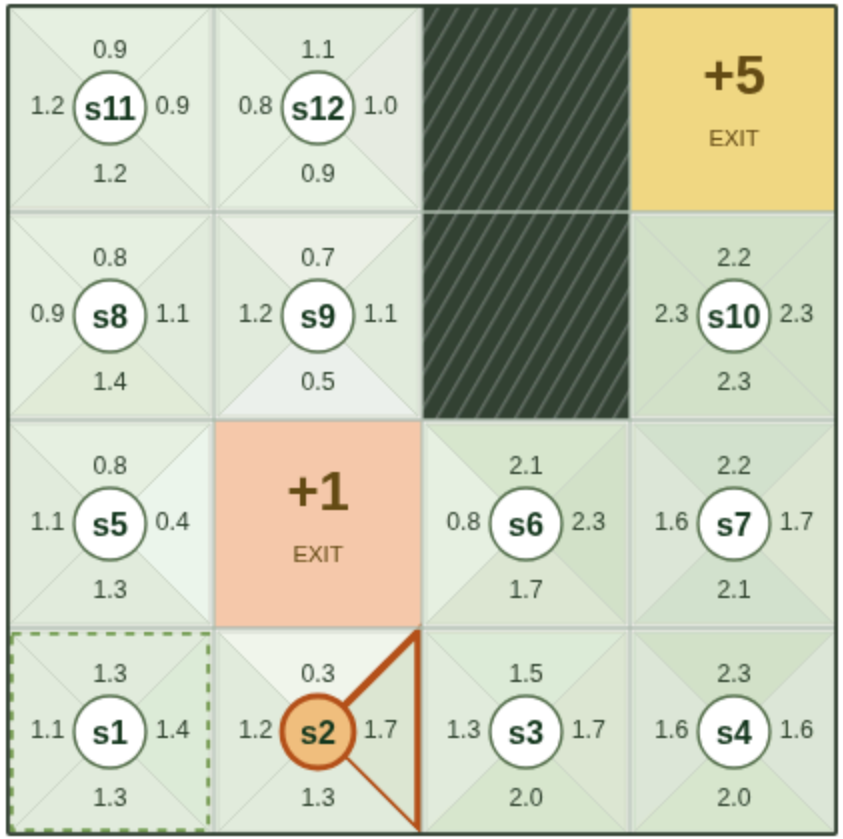
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t) \right]$$

The next action may be exploratory, but the **target assumes the best action** is selected next.

Q-Learning: update the same maze, a different target

In the **same Maze environment**, each state contains four action values. **Click a state or directional triangle**, observe the transition, then update one Q-value.

4×4 MAZE · 4 ACTION VALUES IN EACH CELL



The highlighted triangle is the selected action.

INTERACTIVE Q-LEARNING BACKUP

S_2 · → $\alpha = 0.5, \gamma = 0.9$

$S_2 \rightarrow \rightarrow S_3$

Previous	$Q(s,a)$	1.677
Observed reward	R	0
Greedy continuation	$\max_{a'} Q(S_{t+1}, a')$	1.959 [max]
TD target		1.763
Updated Q(s,a)		1.720

Apply Q-Learning update **Reset Q**

0 updates made. Choose a cell and one of its four action-value triangles.

Q-Learning in six lines

OFF-POLICY TD CONTROL

Initialize $Q(s,a)$ arbitrarily; choose α, γ, ε
For each episode:
  Initialize S
  Repeat:
    Choose $A \sim \varepsilon\text{-greedy}(Q(S,\cdot))$
    Take A, observe R and 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

Define $\max_{a'} Q(\text{terminal},a') = 0$.

Which line is no longer needed compared with SARSA?

We do not need to sample A' before the update.

One slide: SARSA vs. Q-Learning

	SARSA	Q-Learning
TD bootstrap	$Q(S', A')$	$\max_{a'} Q(S', a')$
Learning type	**On-policy**	**Off-policy**
Evaluates	Actual exploratory policy	Greedy target policy
Can use old exploratory samples?	Not in the same straightforward form	Yes, under suitable coverage
With persistent exploration	Accounts for risk of exploratory actions	Plans as though next action were greedy

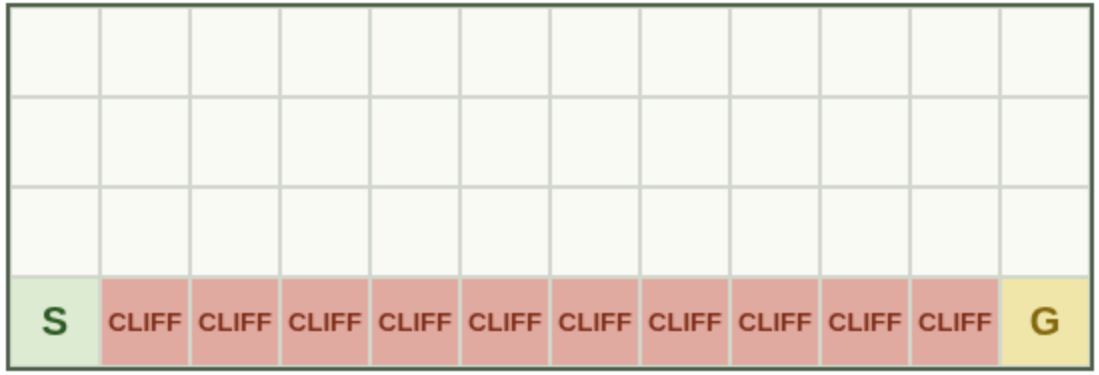
Both are **model-free**, use **one-step TD**, and update **Q(s,a) after every transition**.

PART 5

Cliff Walking

A small world where an exploratory mistake can be expensive.

Cliff Walking: the setup



Start S Goal G Cliff: -100

- Moves up, down, left, right.
- Each normal step yields **-1**.
- Walking into the cliff yields **−100 and resets to S** (the episode continues).
- Only reaching **G** terminates an episode.
- Undiscounted episodic task: γ **= 1**.
- Compare both algorithms with ε **= 0.1**.

PREDICT

Which learned route stays farther away from the cliff when exploration continues?

Textbook convention, Sutton & Barto (2018), Example 6.6: the cliff resets to S without terminating the episode.

Cliff Walking: train both agents

Both agents: tabular, $\alpha = 0.5$, $\gamma = 1$

Each train click runs actual TD updates in your browser

Exploration ε 0.10 +50 ep +300 +1000 Reset

SARSA · ON-POLICY

·	·	·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	·	·	·
S	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	G

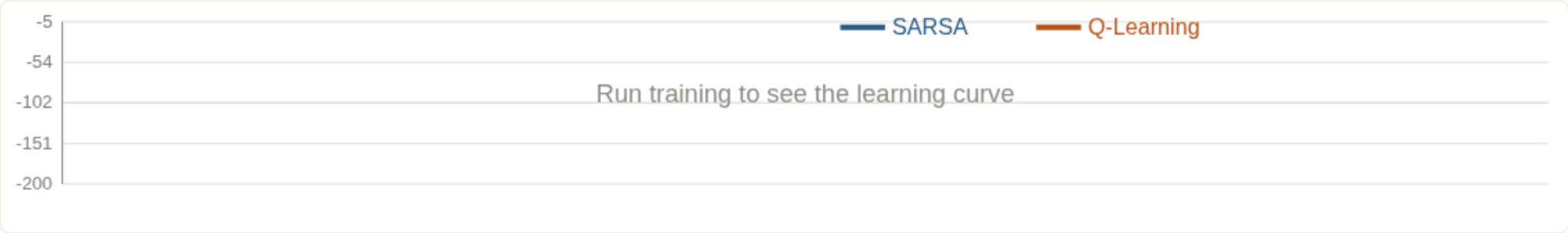
0 episodes

Q-LEARNING · OFF-POLICY

·	·	·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	·	·	·
S	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	CLIFF	G

0 episodes

EPISODE RETURN · 30-EPISODE MOVING AVERAGE (HIGHER IS BETTER)



Why the two algorithms can behave differently

SARSA

Its TD target uses the next action actually chosen by the ε-greedy policy.

It learns that walking next to the cliff makes **future exploration risky**.

Often prefers a **safer route** while exploration persists.

Q-LEARNING

Its TD target assumes the next action is $\arg\max_a Q(S', a)$.

It learns the **shortest greedy route**, which can pass near the cliff.

But its exploratory behaviour can still slip into the cliff.

Thought experiment: what if ε = 0?

EXERCISE 6.12 · SUTTON & BARTO

Suppose both algorithms always select greedily. Are SARSA and Q-Learning now **exactly the same**? Will their updates and trajectories be identical?

ANSWER

If both start with the **same Q**, see the **same transitions**, and break greedy ties **the same way**, then $Q(S', A') = \max_a Q(S', a)$ and their updates coincide.

Different tie-breaking, random seeds or sampled trajectories can still lead to different runs.

PART 6

Extensions & takeaways

Convergence, reusing experience, and Double Q-Learning.

When do we converge?

For **finite tabular MDPs**, convergence results rely on conditions — not just the update formula.

BOTH METHODS

- Every relevant pair (s, a) is visited sufficiently often.
- Learning rates decrease appropriately: $\sum_t \alpha_t = \infty, \quad \sum_t \alpha_t^2 < \infty.$

POLICY CONDITIONS

- **SARSA:** a GLIE-style policy that becomes greedy in the limit.
- **Q-Learning:** off-policy sampling must still cover all relevant pairs sufficiently.

A constant α and constant ε can be useful in demonstrations and nonstationary settings, but do **not** satisfy these exact tabular convergence conditions.

Off-policy learning reuses experience

Yesterday
Random exploratory agent

Today
Updated target policy

Tomorrow
More transitions

All these transitions can inform Q-Learning, if the state-action coverage is appropriate.

- Behaviour policy: where did this transition come from?
- Target policy: which future value is in the bootstrap target?
- With Q-Learning, these do **not have to match**.
- This motivates **experience replay** and later deep RL variants.

The Double Q-Learning idea

The problem: taking the maximum of noisy Q estimates can systematically **overestimate action values**.

Maintain **two** action-value tables, Q_1 and Q_2:

RANDOMLY UPDATE ONE TABLE

Choose the best next action **with one table**, but evaluate it **using the other**.

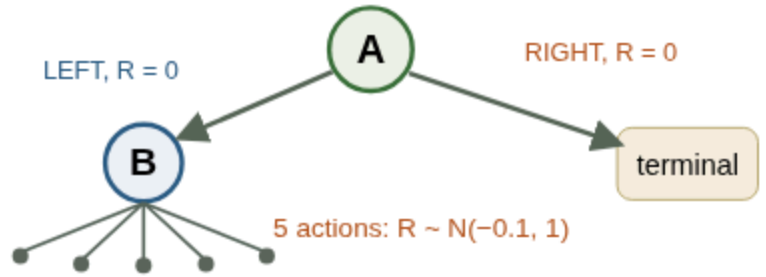
$$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\big[R + \gamma Q_2(S', \arg\max_a Q_1(S',a)) - Q_1(S,A)\big]$$

$$Q_2(S,A) \leftarrow Q_2(S,A) + \alpha\big[R + \gamma Q_1(S', \arg\max_a Q_2(S',a)) - Q_2(S,A)\big]$$

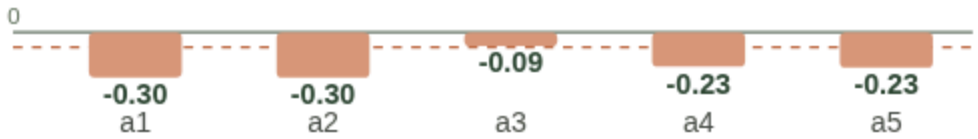
Use one of the two updates on each transition, with terminal bootstrap set to zero. The behaviour policy can be ε-greedy w.r.t. $Q_1 + Q_2$.

Double Q-Learning: maximisation bias, then correction

Sutton & Barto, Fig. 6.5 (simplified to five actions at B): in state **A**, **RIGHT** terminates at reward **0**; **LEFT** leads to **B**, where noisy terminal rewards have mean **−0.1**.



RUN 1 · CURRENTLY LEARNED Q(B,a) FOR 5 ACTIONS · TRUE MEAN −0.1 (DASHED)

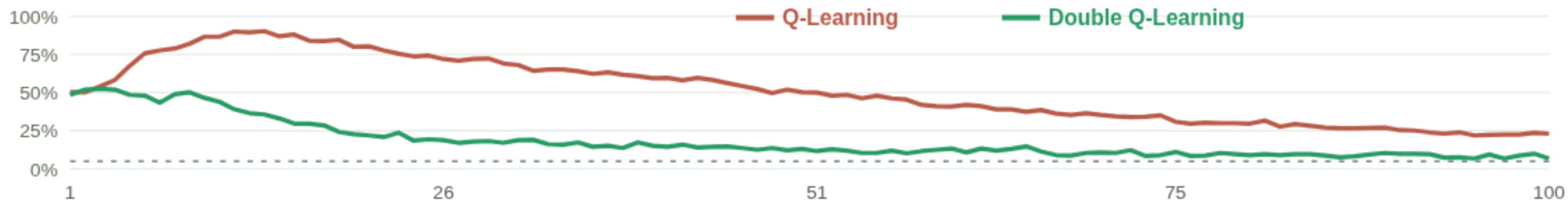


+1 ep | +25 | +100 | +500 | Reset

Q-learning: $Q(A,left)$	Double: $Q(A,left)$
0.005	-0.014
True value of LEFT	True value of RIGHT
−0.100	0.000

100 episodes (350 parallel independent runs). QL in B sampled a3 with R=0.96; Q(B,a) -0.21 → -0.09.
QL A-left TD bootstrap: max stored Q(B,·) = -0.206; **Double:** independent-table bootstrap = 0.000.

PERCENTAGE OF LEFT ACTIONS FROM A · AVERAGE ACROSS INDEPENDENT TRAINING RUNS



When the Q table is too large

For tabular control we store

$$|\mathcal{S}| \times |\mathcal{A}|$$

numbers. This becomes impossible in many real environments.

> Replace the table $Q(s, a)$ with a parameterized estimate $Q_\theta(s, a)$ and update it using sampled TD errors.

MDP	known model
MC / TD	prediction
SARSA / Q-Learning	control
Function approximation	large state spaces
DQN	deep features

What distinguishes the methods?

	Monte Carlo	SARSA	Q-Learning
When update?	After episode	After each step	After each step
Target	G_t	$R + \gamma Q(S', A')$	$R + \gamma \max_{a'} Q(S', a')$
Bootstrapping?	No	Yes	Yes
Policy relationship	On-policy MC control variant	On-policy	Off-policy
Main complication	High return variance	Exploratory target	Max bias, off-policy instability

> In every case, exploration determines which state-action pairs the agent can learn about.

Three questions to check understanding

1. Why is $\pi'(s) = \arg\max_a V(s)$ not a useful model-free policy improvement rule?

Because $V(s)$ does not depend on action a. Without a model we need $Q(s, a)$ to compare actions.

2. What happens in a SARSA update when the next action was exploratory?

SARSA bootstraps from **that exploratory next action's Q-value**, not the maximum.

3. Can Q-Learning behave non-greedily while learning a greedy target?

Yes. It can collect data using ε-greedy behaviour and still bootstrap from $\max_{a'} Q(S', a')$.

Takeaways

01 Prediction → control

Learn action values and improve the policy while collecting data.

02 Exploration matters

ε-greedy prevents a learner from trusting its early guesses forever.

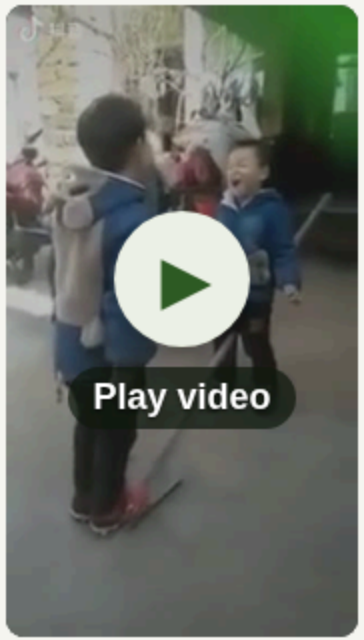
03 SARSA is on-policy

It bootstraps from the next action actually taken.

04 Q-Learning is off-policy

It bootstraps from the best next action, even while exploring.

FINAL VIDEO



MAIN REFERENCES

Sutton & Barto (2018), *Reinforcement Learning: An Introduction*, Chapters 5–6.
David Silver, UCL RL course, Lecture 5: *Model-Free Control*.

HANDS-ON

Karpathy's Gridworld TD demo (online)