Session 3.3

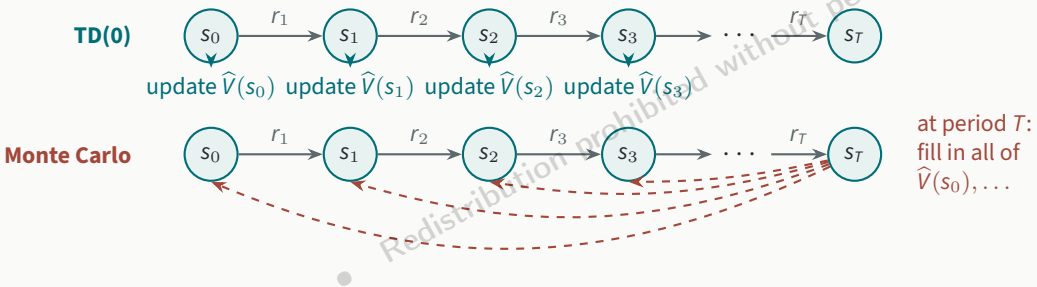
Temporal Difference: TD(0), SARSA, Q-Learning



Zhigang Feng

© Zhigang Feng
Redistribution prohibited without permission

Monte Carlo Waits for the Whole Path. Temporal Difference Does Not



Redistribution prohibited without permission

© Zhaonan Feng

- Deck 3.2 averaged whole returns, $T = 300$: each sample waits T periods and carries every shock on the way.
- **This deck takes $T = 1$: one real reward, and the current estimate for everything after.**

Session 3.3

Evaluation, One Step at a Time

The incremental mean, TD(0), and everything between one step and the whole path



© Zhigang Feng · Redistribution prohibited without permission

The Sample Mean, Written One Observation at a Time

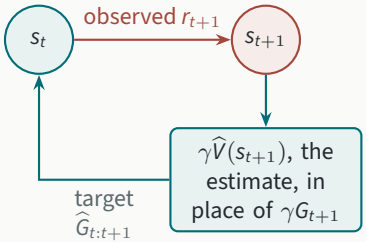
- **The question.** Deck 3.2 averaged returns. How is an average kept up to date as the draws arrive?
- The mean of $n+1$ draws is the old mean plus a correction:

$$\bar{X}_{n+1} = \bar{X}_n + \frac{1}{n+1}\left(X_{n+1} - \bar{X}_n\right).$$

  New estimate $=$ old estimate $+$ step size $\times$ surprise: adaptive expectations with gain $1/(n+1)$.
- Any **step size** η_n will do if it shrinks, but not too fast (the Robbins–Monro conditions): the estimate then settles on the mean.
- A constant η never settles: the estimate hovers around the mean, forgetting old draws at rate $1-\eta$.
- **Forgetting is a virtue when the target itself moves, as it will once the rule is being improved.**

© Zhiguang Feng
Redistribution prohibited without permission

TD(0): One Real Reward, the Rest on Credit



- **The question.** A return waits for the whole path. Can one period teach something?
- Yes: borrow the estimate for the unseen rest. The **TD target** $\widehat{G}_{t:t+1} = r_{t+1} + \gamma \widehat{V}(s_{t+1})$ is the Bellman equation on one draw.
- Nudge the stored value toward it, $\widehat{V}(s_t) \leftarrow \widehat{V}(s_t) + \eta\, \delta_t$, where the **TD error** $\delta_t = \widehat{G}_{t:t+1} - \widehat{V}(s_t)$ is the miss.
- Leaning on your own estimate is **bootstrapping**: one real reward, the rest on credit.
- **Against DP: no sum over z'. Against Monte Carlo: no waiting.**

© Zhigang Feng
Reproduction prohibited without permission

TD(0) Policy Evaluation

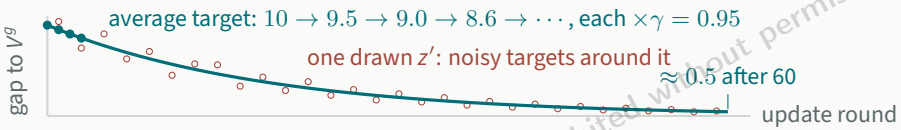
TD(0): evaluate a given policy g along one simulated path

initialise $\widehat{V}(s)$; visit counts $C(s) \leftarrow 0$; $s \leftarrow s_0$
for $t = 0, 1, 2, \ldots$ up to the step budget:
  draw $a \sim g(\cdot \mid s)$; the simulator returns r and s'
  $C(s) \leftarrow C(s) + 1$
  $\delta \leftarrow r + \gamma \widehat{V}(s') - \widehat{V}(s)$
  $\widehat{V}(s) \leftarrow \widehat{V}(s) + \eta\, \delta$ η constant, or shrinking with $C(s)$
  $s \leftarrow s'$

- It converges to V^g if every state keeps being visited and the step shrinks at the right pace; nothing in that says how fast.
- The plain average, $\eta = 1/C(s)$, qualifies and is unusably slow (p. 12).
- **No episodes are needed: the path just runs. Cutting it is not ending it.**

© Thierry Post · Redistribution prohibited without permission

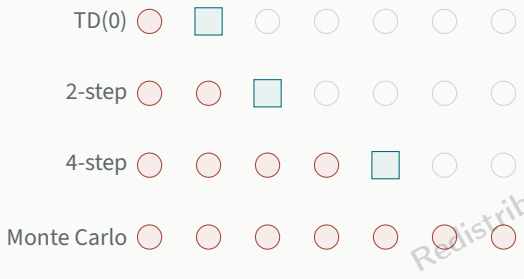
Why Tabular TD(0) Converges: the Target Contracts



- **The question.** The target $r + \gamma\widehat{V}(s')$ leans on a guess. Why does that not go astray?
- **The target is less wrong than its guess.** r is real; only tomorrow is guessed, and it is discounted. A guess off by 10 everywhere gives a target off by $0.95 \times 10 = 9.5$: on average, one step of value iteration for g.
- **The noise averages out.** One drawn z' scatters the target around that average; a shrinking step, $\eta = C(s)^{-0.6}$, lets no single draw move the estimate far.
- **Every state keeps being visited.** A cell is fixed only when the path passes through it; the trembles see to that.
- **TD is noisy value iteration, one state at a time. Cells share nothing, so fixing one never spoils another; functions that share lose the guarantee (Deck 3.4).**

© Lilian Feng · Redistribution prohibited without permission

n-Step Returns: Between One Step and the Whole Path



© Zhigang Feng
Redistribution prohibited without permission

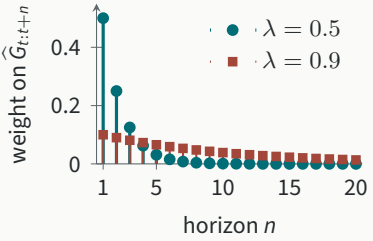
- **The question.** How many real periods before the estimate? In the lab, five beat one with the branches summed, not with one drawn.
- The ***n*-step return**: n real rewards, then the estimate,

$$\widehat{G}_{t:t+n} = \sum_{j<n} \gamma^j r_{t+j+1} + \gamma^n \widehat{V}(s_{t+n}).$$

- $n = 1$ is TD(0); large n is Monte Carlo. Larger n: less bias, more variance (why: p. 10).
- **And the update must wait n periods.**

Source: Session 3 lab, block B.

TD(λ): Average Every Horizon, Credit the Recent States



- **The question**: n-step returns make us pick one n. Why not average every horizon? **TD(λ)** does, with weights that fade:

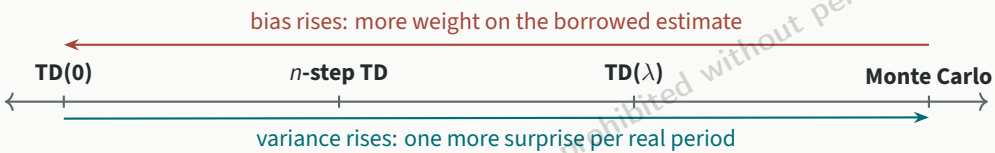
$$\widehat{G}_t^{\lambda} = (1-\lambda)\sum_{n\geq 1}\lambda^{n-1}\widehat{G}_{t:t+n}.$$

$\lambda = 0$: TD(0)'s target. $\lambda \to 1$: Monte Carlo's whole return.

- It can be computed as the path runs: when today's surprise δ_t arrives, credit it to the states visited lately, a state k periods back taking the share $(\gamma\lambda)^k$.
- The **eligibility trace** is the bookkeeping that remembers those shares; summed over the path, the updates are the same average over horizons.
- **TD(λ) charges today's surprise to the states that led here, recent ones most: credit assignment, which choice earned the reward.**

© Zhiguo Feng Redistribution prohibited without permission

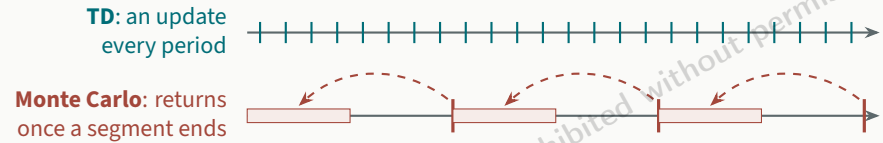
One Dial: Bias against Variance



© Zhang Fang. Redistribution prohibited without permission

- **The question.** Why does a longer target trade bias for variance?
- A target's error has two parts. Each real period adds one surprise, a shock or a tremble, mean zero and fresh every period: that is noise, and n periods add n of it.
- The borrowed estimate $\gamma^n \widehat{V}(s_{t+n})$ is the only part that can be wrong on average: that is bias, all of it in TD(0), none in Monte Carlo, discounted by γ^n in between.
- **The bias shrinks as $\widehat{V}$ learns; the variance does not. Early, lean on real rewards; later, on the estimate.**

TD against Monte Carlo on One Long Path: Bias Early, Variance Late



© Yiyang Feng · Redistribution prohibited without permission

- **The set-up.** One fixed exploring rule, one long path, no resets, as an economy produces data. TD uses each transition at once; Monte Carlo cuts the path into segments and waits.
- **Early**, Monte Carlo wins: its returns hold no guess, so its first segments set the right level. TD's targets are mostly the starting guess, fading one period back per update.
- **Later**, TD wins: the borrowed error is small, and only noise remains: one surprise in TD's target, a whole segment's in Monte Carlo's.
- How fast the guess fades is the step size's doing (p. 12).
- **Bias fades as the estimate learns; variance does not. That decides who leads.**

Target and Step Size Cannot Be Judged Apart

- **The question.** TD converges with the plain average $\eta = 1/C(s)$. Why is that step unusably slow?
- The step says how much each target counts. The plain average counts every target equally, which suits Monte Carlo: its returns are equally good from the first.
- TD's early targets rest on the starting guess. Counted like the later, better ones, the guess fades so slowly that halving its error takes more visits than any budget allows.
- A step that shrinks more slowly than $1/C(s)$ lets go of early targets while still settling: far faster, and still convergent.
- The start matters to TD alone. Steady-state utility for ever is a sensible guess; under the plain average, only a start near the truth ends near it.
- **With a bootstrapped target, choose the target, the step size and the start together.**

© Zhirang Feng · Redistribution prohibited without permission

Session 3.3

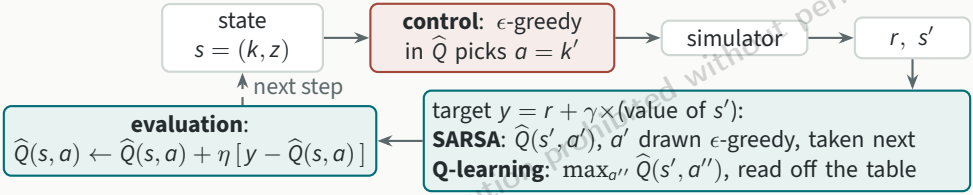
Control: SARSA and Q-Learning

Improve the rule at every step, and ask whose value is being learned



© Zhigang Feng Redistribution prohibited without permission

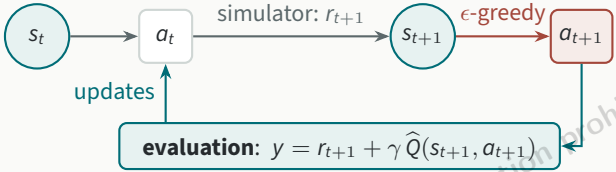
From Evaluation to Control: Improve at Every Step



- **Control** improves the rule (Deck 3.2): act **ϵ-greedily** in the latest $\widehat{Q}$, the best-looking k' usually, a random feasible one now and then.
- **Evaluation**: one TD update of $\widehat{Q}(s, a)$, the value of taking a in s, toward the target y: p. 5's one-step target with $\widehat{Q}$ for $\widehat{V}$.
- Both see the same r and s'; the targets differ only after an exploring draw.
- **SARSA values tomorrow by the rule it follows, Q-learning by the greedy rule.**

© Zhirang Feng · Redistribution prohibited without permission

SARSA, Step by Step



control: drawn from the rule in the current $\widehat{Q}$, and really taken next

- In s_t, take a_t, chosen a period earlier; the simulator returns r_{t+1} and s_{t+1}.
- **Control**: in s_{t+1}, draw a_{t+1} ϵ-greedily in the current $\widehat{Q}$. It is the action the economy will really take next.
- **Evaluation**: $\widehat{Q}(s_t, a_t) \leftarrow \widehat{Q}(s_t, a_t) + \eta \left[r_{t+1} + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t) \right]$.
- Move on, $(s_t, a_t) \leftarrow (s_{t+1}, a_{t+1})$. Each step uses (s, a, r, s', a'): hence the name, **SARSA**.
- **One rule does both jobs: it chooses the actions, and it is the rule being valued.**

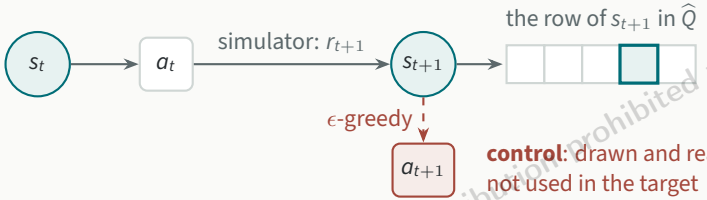
© Zhigang Feng
No distribution prohibited without permission

What SARSA Learns, and Where Its Control Ends

- **The question.** SARSA improves the rule at every step. Whose value does it learn, and does it reach g^*?
- The value of the rule making the data, trembles included (**on-policy**). A trembling saver wastes capital tomorrow, so it learns to save a little less than g^* would.
- With ϵ fixed it settles on the best **ϵ-soft** rule, one keeping every action at probability $\geq \epsilon/|\mathcal{A}(s)|$: the best way to tremble, not g^*.
- **GLIE**, greedy in the limit with infinite exploration: let ϵ fall to zero slowly, every pair still tried; then SARSA reaches g^*.
- **Expected SARSA** averages the target over a' instead of drawing it: same mean, less noise.
- **SARSA learns the value of exploring, and improves the rule it follows.**

© Zhigang Feng · Redistribution prohibited without permission

Q-Learning, Step by Step



evaluation: read off its max; no action is tried

control: drawn and really taken next; not used in the target

© Zhirang Feng • Redistribution prohibited without permission

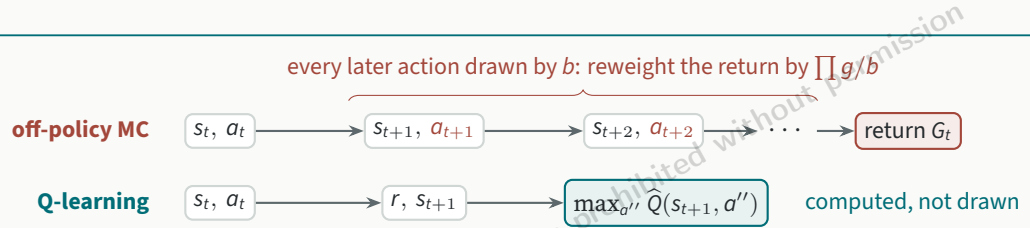
- The transition is SARSA's: in s_t take a_t; the simulator returns r_{t+1} and s_{t+1}.
- **Evaluation**: $\widehat{Q}(s_t, a_t) \leftarrow \widehat{Q}(s_t, a_t) + \eta \left[r_{t+1} + \gamma \max_{a'' \in \mathcal{A}(s_{t+1})} \widehat{Q}(s_{t+1}, a'') - \widehat{Q}(s_t, a_t) \right]$. No action is tried: the max reads numbers already in the table.
- **Control**: in s_{t+1}, draw a_{t+1} ϵ-greedily and take it. It drives the path; the target ignores it.
- **Two rules, two jobs: the ϵ-greedy rule acts, and the greedy rule is valued.**

What Q-Learning Learns: Q^*, from Exploring Data

- **The question.** Q-learning's target reads a $\max$ off the table instead of the action taken. What does that buy?
- Value function iteration, done one sampled transition at a time: with the expectation exact and $\eta = 1$, the update is Deck 3.1's value iteration. The improvement sits inside the target.
- So it values the greedy **target rule** from the data of the ϵ-greedy **behaviour rule**: **off-policy**. A fixed $\epsilon > 0$ suffices; deploy the greedy rule, exploring was for learning.
- Three costs remain: cells rarely visited stay wrong; the $\max$ over noisy estimates is biased up (p. 24); welfare while learning is the trembling rule's.
- **Off-policy buys the right target, not the right data.**

© Thiago Pinto. Redistribution prohibited without permission

Why Q-Learning Needs No Importance Ratio



© Zhiyang Feng · Reproduction prohibited without permission

- **The question.** Q-learning values the greedy rule g from the exploring rule b's data, yet Deck 3.2's off-policy Monte Carlo had to reweight each return by $\prod g/b$. Why not here?
- Who chose each part of the target? a_t is given, the update is for that cell; r and s_{t+1} are the economy's under any rule; the $\max$ is computed for g.
- Nothing b drew after a_t enters the target, so there is nothing to reweight. A Monte Carlo return holds every later action, each drawn by b: hence the product.
- **One step, no ratio: b only decides which cells get data. A target holding more of the walked path would bring the ratio back.**

One Algorithm, One Line of Difference

TD control: SARSA and Q-learning

1 initialise $\widehat{Q}$ on feasible cells, counts $C \leftarrow 0$; $s \leftarrow s_0$; choose a, ϵ-greedy in $\widehat{Q}$ — **control**

2 **for** $t = 0, 1, 2, \ldots$ up to the step budget:

3 take a; the simulator returns r and s'

4 choose a' in $\mathcal{A}(s')$, ϵ-greedy in $\widehat{Q}$: the action taken next — **control**

5 **SARSA:** $y \leftarrow r + \gamma\, \widehat{Q}(s', a')$ **Q-learning:** $y \leftarrow r + \gamma \max_{a'' \in \mathcal{A}(s')} \widehat{Q}(s', a'')$ — **evaluation**

6 $C(s,a) \leftarrow C(s,a) + 1$; $\widehat{Q}(s,a) \leftarrow \widehat{Q}(s,a) + C(s,a)^{-0.6}[\, y - \widehat{Q}(s,a)\,]$ — **evaluation**

7 $(s, a) \leftarrow (s', a')$

- **Control** (lines 1, 4) is the same in both: ϵ-greedy in the current $\widehat{Q}$, which line 6 updates.
- **Evaluation** (lines 5–6): SARSA values the a' that line 4 drew, so one rule acts and is valued (on-policy); Q-learning values the greedy choice, and a' only acts (off-policy).
- **Line 5 is the whole difference. Deploy $g(s) = \arg\max_a \widehat{Q}(s,a)$ in both.**

Source: Session 3 lab, block E.

© Eugene Yan Redistribution prohibited without permission

Session 3.3

Using Samples Well

Q-learning reaches Q^* in the limit. On a budget of steps, where do samples go to waste, and how is each waste cut?



© Zhigang Feng · Redistribution prohibited without permission

Q-Learning on a Budget: Four Places Samples Go to Waste

2 how much it counts: the step size (p. 12)

3 a max over noisy estimates (p. 24)

$$\widehat{Q}(s,a) \leftarrow \widehat{Q}(s,a) + \eta_{C(s,a)} \left[r + \gamma \max_{a''} \widehat{Q}(s',a'') - \widehat{Q}(s,a) \right]$$

1 which (s,a) gets the sample: the behaviour rule (p. 23)

4 one real period, then the estimate (p. 8–10)

© Zhaohong Wang · Redistribution prohibited without permission

- **The question.** Convergence asks for every pair visited for ever and a shrinking step, but not *how many* steps. On a budget, each part of the update can waste samples.
- Two of the four were met on the evaluation side: the step size, and how far the target looks. The other two are about the data and the max: next.
- **Every fix changes where samples come from, or how the target uses them.**

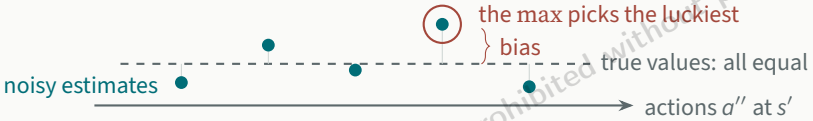
Which Cells Get Data: Explore Where It Can Pay



© Zheng Tang · Redistribution prohibited without permission

- **The problem.** Q-learning learns Q^* for any fixed $\epsilon > 0$, but only in cells with data, and ϵ-greedy spreads its trials evenly: a hopeless k' gets as many as the runner-up.
- **Boltzmann exploration**: a logit choice rule, $g(a \mid s) \propto \exp\{\widehat{Q}(s,a)/\tau\}$ with temperature τ; the worse an action looks, the fewer its trials.
- **Optimistic initial values**: start $\widehat{Q}$ above the values it will learn; an untried action looks good until tried.
- **Pace**: ϵ or τ may fall with the counts, as long as every pair keeps being tried.
- **Explore near the best, and where you have not looked.**

What the Max Adds: Maximisation Bias, and a Second Table



- **The problem.** The target's $\max$ is over noisy estimates, and $E[\max \widehat{Q}] \geq \max E[\widehat{Q}]$: the action that looks best is partly the luckiest (Deck 3.2's winner's curse).
- Bootstrapping passes the inflated value back to earlier states. SARSA is not immune: its a' is the greedy action most of the time.
- **The fix**, **double Q-learning**: keep two tables and update one of them, chosen at random, each step. One table picks the winner, the other prices it; act on their sum.
- **Luck that picks an action no longer lifts its value.**

© Zhigang Feng — Redistribution prohibited without permission

The Backup Spectrum: Two Choices Define an Evaluation Method



- **The question.** Every evaluation in Decks 3.1–3.3 solves $V^g = T^g V^g$. What differs is the **backup**, the target used.
- **Width**: sum every branch, exact but reading P_z, or draw one, no P_z but one draw's noise. DP sums; this deck samples: we pretend not to know P_z.
- **Depth**: lean on the stored $\widehat{V}$ after one period, quick but biased while it is poor, or run to the end, honest but noisy and slow; n, λ, step and start tune it.
- **Every point on the map is the same Bellman equation.**

© Zhiguo Feng Unauthorized distribution prohibited without permission