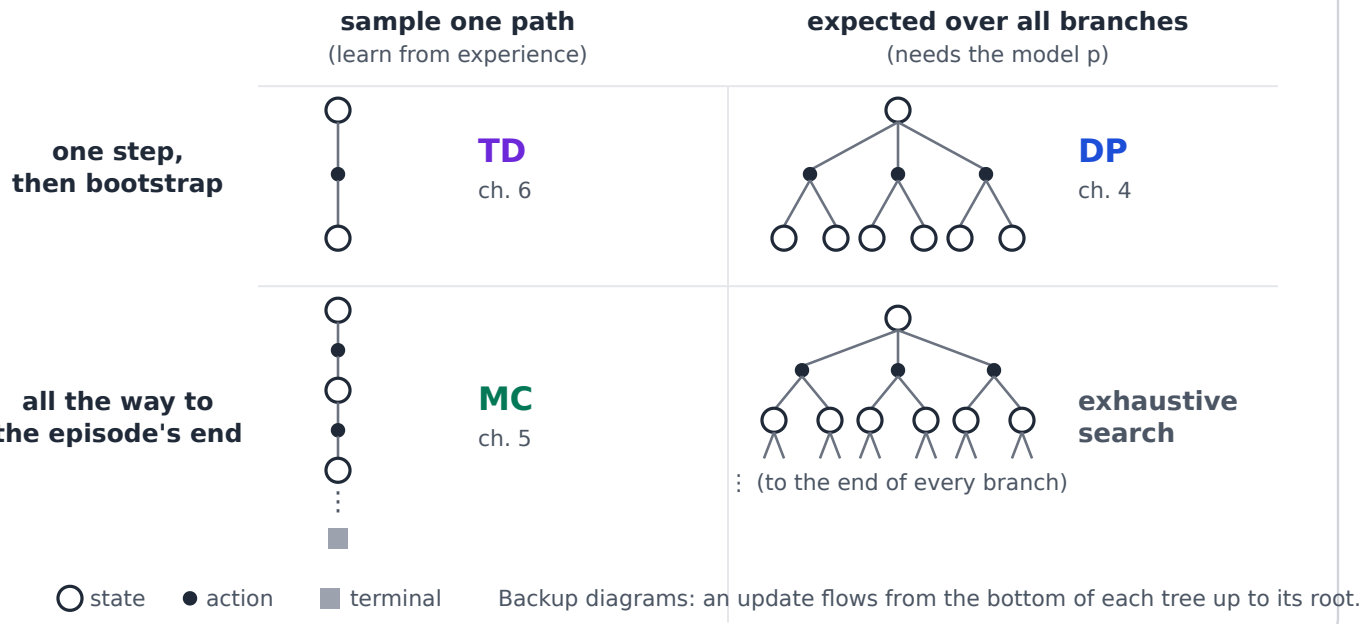


Tabular RL cheat sheet · DP, MC and TD (Sutton & Barto, ch. 4-6)

Equation numbers match the book (2nd ed.). Algorithms are condensed paraphrases of the book's boxes. $p(s', r \mid s, a)$ is the model; π is the policy.

The big picture: two ways a target can differ



Chapter 4 · Dynamic Programming

Needs the full model $p(s', r \mid s, a)$. Sweeps over states with expected updates; no sampling, but it bootstraps.

4.1-4.4 · Formulas

Bellman for v_π (4.3)	$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
policy evaluation (4.5)	$v_{k+1}(s) = \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_k(s')]$
same, for q_π	$q_{k+1}(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') q_k(s', a') \right]$
q from v (4.6)	$q_\pi(s, a) = \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_\pi(s')]$
improvement thm (4.7)	$q_\pi(s, \pi'(s)) \geq v_\pi(s) \text{ for all } s$
⇒ (4.8)	$v_{\pi'}(s) \geq v_\pi(s) \text{ for all } s$
greedy policy (4.9)	$\pi'(s) = \arg \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_\pi(s')]$
value iteration (4.10)	$v_{k+1}(s) = \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_k(s')]$
same, for q_*	$q_{k+1}(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_k(s', a') \right]$
If greedy improvement leaves π unchanged, v_π satisfies the Bellman optimality equation (4.1): π is optimal. $v_k \rightarrow v_\pi$ (and value iteration $\rightarrow v_*$) whenever $\gamma < 1$ or every episode is guaranteed to terminate.	

4.5-4.7 · Ideas without new formulas

Asynchronous DP: update states in any order, one at a time and in place, as long as every state keeps getting updated. Lets you focus on relevant states or learn while acting.
Generalized policy iteration (GPI): policy evaluation and policy improvement interact at any granularity. Almost every RL method (MC, TD) is GPI with approximate evaluation.
Efficiency: polynomial in $ S $ and $ A $, versus $ A ^{ S }$ deterministic policies for direct search. The catch is the curse of dimensionality: $ S $ grows exponentially with the number of state variables.
Linear system: (4.4) is $ S $ equations in $ S $ unknowns, solvable directly; iteration scales better.

Iterative policy evaluation

§4.1 · $V \approx v_\pi$

Input: policy π ; threshold $\theta > 0$ (a small number)
Init: $V(s)$ arbitrary for every s , with $V(\text{terminal}) = 0$
Repeat (one sweep):
 $\Delta \leftarrow 0$
 for each state s :
 $v \leftarrow V(s)$
 $V(s) \leftarrow \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$
 $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$
In place (one array): new values are used as soon as they're computed. Usually converges faster.

Policy iteration

§4.3 · $\pi \approx \pi_*$

1. Init: $V(s)$ and a deterministic $\pi(s)$ arbitrary for every s
2. Evaluate: run iterative policy evaluation for the current π , using $a = \pi(s)$:
 $V(s) \leftarrow \sum_{s', r} p(s', r \mid s, \pi(s)) [r + \gamma V(s')]$ until $\Delta < \theta$
3. Improve: $\text{stable} \leftarrow \text{true}$
 for each state s :
 $\text{old} \leftarrow \pi(s)$
 $\pi(s) \leftarrow \arg \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$
 if $\text{old} \neq \pi(s)$: $\text{stable} \leftarrow \text{false}$
 if stable : stop and return $V \approx v_*, \pi \approx \pi_*$; **else** go to 2
Break argmax ties consistently, or the loop can switch forever between equally good policies.

Value iteration

§4.4 · $\pi \approx \pi_*$

Input: threshold $\theta > 0$
Init: $V(s)$ arbitrary for every s , with $V(\text{terminal}) = 0$
Repeat:
 $\Delta \leftarrow 0$
 for each state s :
 $v \leftarrow V(s)$
 $V(s) \leftarrow \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$
 $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$
Output $\pi(s) = \arg \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$

One sweep of evaluation plus improvement at once: the max replaces the average over π .

Chapter 5 · Monte Carlo methods

No model: learn from complete sampled episodes. Targets are actual returns, so no bootstrapping; episodic tasks only.

5.1-5.4 · Prediction and on-policy control

MC estimate	$V(s) = \frac{1}{ \mathcal{T}(s) } \sum_{t \in \mathcal{T}(s)} G_t$ (average of observed returns)
visits counted	$\mathcal{T}(s)$: first visits per episode (first-visit) or all visits (every-visit)
incremental mean	$V(s) \leftarrow V(s) + \frac{1}{N(s)}[G_t - V(s)]$
constant- α MC (6.1)	$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
greedy policy (5.1)	$\pi(s) = \arg \max_a q(s, a)$
why it improves	$q_{\pi_k}(s, \pi_{k+1}(s)) = \max_a q_{\pi_k}(s, a) \geq v_{\pi_k}(s)$
ϵ -soft	$\pi(a \mid s) \geq \epsilon / \mathcal{A}(s) $ for all s, a
ϵ -greedy	$\pi(a \mid s) = 1 - \epsilon + \frac{\epsilon}{ \mathcal{A}(s) }$ if $a = A^*$, else $\frac{\epsilon}{ \mathcal{A}(s) }$

ϵ -soft improvement (5.2): π' ϵ -greedy w.r.t. $q_{\pi'}$, π any ϵ -soft policy

$$\sum_a \pi'(a \mid s) q_{\pi}(s, a) = \frac{\epsilon}{|\mathcal{A}|} \sum_a q_{\pi}(s, a) + (1 - \epsilon) \max_a q_{\pi}(s, a) \geq \sum_a \pi(a \mid s) q_{\pi}(s, a) = v_{\pi}(s)$$

The $\geq$ holds because the max beats any weighted average with weights $(\pi(a|s) - \epsilon/|A|)/(1 - \epsilon) \geq 0$.
Result: converges to the best ϵ -soft policy, not the truly optimal one (decay ϵ for that).

5.5-5.6 · Off-policy prediction by importance sampling

coverage	$\pi(a \mid s) > 0 \Rightarrow b(a \mid s) > 0$
ratio (5.3)	$\rho_{t:T-1} = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}$ (the dynamics cancel)
corrects returns (5.4)	$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_{\pi}(s)$
ordinary IS (5.5)	$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{ \mathcal{T}(s) }$
weighted IS (5.6)	$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$
incremental (5.8)	$V_{n+1} = V_n + \frac{W_n}{C_n}[G_n - V_n], \quad C_{n+1} = C_n + W_{n+1}, \quad C_0 = 0$

Ordinary IS: unbiased (first-visit), but its variance can be infinite.
Weighted IS: biased (the bias $\rightarrow 0$ with more data), much lower variance; usually preferred.

5.8-5.9 · Starred sections (lower-variance off-policy)

flat partial return	$\bar{G}_{t:h} = R_{t+1} + R_{t+2} + \dots + R_h$
discounting-aware: split G_t into flat partial returns, then use ratios only up to each horizon h	$G_t = (1 - \gamma) \sum_{h=t+1}^{T-1} \gamma^{h-t-1} \bar{G}_{t:h} + \gamma^{T-t-1} \bar{G}_{t:T}$
per-decision (5.14)	$\mathbb{E}[\rho_{t:T-1} R_{t+1}] = \mathbb{E}[\rho_{t:t} R_{t+1}]$
per-decision return, used in $V(s) = \Sigma \tilde{G}_t / \mathcal{T}(s) $ (5.15)	$\tilde{G}_t = \rho_{t:t} R_{t+1} + \gamma \rho_{t:t+1} R_{t+2} + \gamma^2 \rho_{t:t+2} R_{t+3} + \dots + \gamma^{T-t-1} \rho_{t:T-1} R_T$

First-visit MC prediction

§5.1 · $V \approx v_{\pi}$

Input: policy π . **Init:** $V(s)$ arbitrary; Returns(s) $\leftarrow$ empty list
Loop for each episode:
 generate an episode with π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$
 $G \leftarrow 0$
 for $t = T-1, T-2, \dots, 0$ (backward):
 $G \leftarrow \gamma G + R_{t+1}$
 if S_t is not among $S_0, \dots, S_{t-1}$ (first visit):
 append G to Returns(S_t); $V(S_t) \leftarrow \text{average}(\text{Returns}(S_t))$

Every-visit MC: drop the first-visit check. Both converge to v_{π} .

Monte Carlo ES (exploring starts)

§5.3 · $\pi \approx \pi_*$

Init: $\pi(s)$ arbitrary; $Q(s, a)$ arbitrary; Returns(s, a) $\leftarrow$ empty list
Loop for each episode:
 pick S_0, A_0 at random so that every pair has probability > 0
 generate an episode from S_0, A_0 , then follow π
 $G \leftarrow 0$; **for** $t = T-1, \dots, 0$:
 $G \leftarrow \gamma G + R_{t+1}$
 if (S_t, A_t) is a first visit in this episode:
 append G to Returns(S_t, A_t); $Q(S_t, A_t) \leftarrow \text{average}$
 $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

On-policy first-visit MC control (ϵ -soft)

§5.4 · $\pi \approx \pi_*$

Input: small $\epsilon > 0$. **Init:** π any ϵ -soft policy; Q arbitrary; Returns empty
Loop for each episode:
 generate an episode following π
 $G \leftarrow 0$; **for** $t = T-1, \dots, 0$:
 $G \leftarrow \gamma G + R_{t+1}$
 if (S_t, A_t) is a first visit:
 append G to Returns(S_t, A_t); $Q(S_t, A_t) \leftarrow \text{average}$
 $A^* \leftarrow \arg\max_a Q(S_t, a)$ (ties broken arbitrarily)
 make π ϵ -greedy at S_t : $1 - \epsilon + \epsilon/|A|$ for A^* , $\epsilon/|A|$ otherwise

Off-policy MC prediction (weighted IS)

§5.6 · $Q \approx q_{\pi}$

Input: target policy π . **Init:** $Q(s, a)$ arbitrary; $C(s, a) \leftarrow 0$
Loop for each episode:
 $b \leftarrow$ any policy with coverage of π ; generate an episode following b
 $G \leftarrow 0$; $W \leftarrow 1$
 for $t = T-1, \dots, 0$, while $W \neq 0$:
 $G \leftarrow \gamma G + R_{t+1}$; $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$

 $W \leftarrow W \cdot \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$

With $\pi = b$, W stays 1 and this is ordinary every-visit MC with 1/n averaging.

Off-policy MC control

§5.7 · $\pi \approx \pi_*$

Init: $Q(s, a)$ arbitrary; $C(s, a) \leftarrow 0$; $\pi(s) \leftarrow \arg\max_a Q(s, a)$
Loop for each episode:
 $b \leftarrow$ any soft policy; generate an episode following b
 $G \leftarrow 0$; $W \leftarrow 1$; **for** $t = T-1, \dots, 0$:
 $G \leftarrow \gamma G + R_{t+1}$; $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$

 $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (ties broken consistently)
 if $A_t \neq \pi(S_t)$: stop this episode's loop
 $W \leftarrow W / b(A_t \mid S_t)$ (π is greedy, so $\pi(A_t|S_t) = 1$)

Learns only from the tails of episodes after the last non-greedy action: can be slow.

Chapter 6 · Temporal-difference learning

No model, like MC; bootstraps, like DP. Updates after every step, so it works online and on continuing tasks.

6.1-6.3 · Prediction

constant-α MC (6.1)	$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
TD(0) (6.2)	$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
MC's target (6.3)	$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
DP's and TD's (6.4)	$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
TD error (6.5)	$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
MC error (6.6)	$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k$ (if V is fixed during the episode)
batch MC converges to	$V(s) = \text{average observed return from } s$
batch TD converges to	$V(s) = \hat{r}(s) + \gamma \sum_{s'} \hat{P}(s' \mid s) V(s')$ (certainty equivalence)
TD(0) converges to v_π : in the mean for a small constant α , and with probability 1 when α decays with $\sum \alpha = \infty$, $\sum \alpha^2 < \infty$. TD is usually faster than constant- α MC on stochastic tasks (not proven).	

6.4-6.7 · Control

Sarsa (6.7)	$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
Q-learning (6.8)	$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
Expected Sarsa (6.9)	$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right]$
Double Q-learning (6.10), with Q1 and Q2 swapped half the time	$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q_2(S_{t+1}, \arg \max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t) \right]$
maximization bias	$\mathbb{E}[\max_a Q(s, a)] \geq \max_a q(s, a)$ if each Q is unbiased but noisy
double estimator	$A^* = \arg \max_a Q_1(a), \quad \mathbb{E}[Q_2(A^*)] = q(A^*)$

Control methods side by side

	next-action term	policy type	learns (fixed ε)	note
Sarsa	$Q(S', A'), A' \text{ taken}$	on-policy	q of ε-greedy	safer (cliff)
Expected Sarsa	$\sum \pi(a S') Q(S', a)$	either	q of π	less variance
Q-learning	$\max Q(S', a)$	off-policy	q*	max bias
Double Q	Q2 at argmax Q1	off-policy	q*	no max bias

Example 6.4: batch MC vs batch TD on eight episodes

Data: A, 0, B, 0 once; B, 1 six times; B, 0 once (γ = 1).
Both: V(B) = 6/8 = 0.75.
Batch MC: V(A) = 0, the one observed return from A (best fit to the returns seen).
Batch TD: V(A) = 0.75, since A always led to B with reward 0 (fits the Markov model).
TD's answer is the certainty-equivalence estimate and usually generalizes better.

Tabular TD(0) §6.1 · $V \approx v_\pi$

Input: policy π; step size α ∈ (0, 1]
Init: V(s) arbitrary for every s, with V(terminal) = 0
Loop for each episode: initialize S
 Loop for each step, until S is terminal:
 A ← action given by π for S; take A, observe R, S'
 $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
 S ← S'

Sarsa (on-policy TD control) §6.4 · $Q \approx q_*$

Input: α ∈ (0, 1], small ε > 0. **Init:** Q arbitrary, Q(terminal, ·) = 0
Loop for each episode: initialize S; choose A from S (ε-greedy in Q)
 Loop for each step, until S is terminal:
 take A, observe R, S'; choose A' from S' (ε-greedy in Q)
 $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
 S ← S'; A ← A' (A' really is the next action taken)
Converges to π*, q* if every pair is visited infinitely often and ε decays to 0 (GLIE).

Q-learning (off-policy TD control) §6.5 · $\pi \approx \pi_*$

Input: α ∈ (0, 1], small ε > 0. **Init:** Q arbitrary, Q(terminal, ·) = 0
Loop for each episode: initialize S
 Loop for each step, until S is terminal:
 choose A from S (ε-greedy in Q); take A, observe R, S'
 $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
 S ← S' (the next action is chosen fresh, after the update)
Learns q* whatever the behavior policy, as long as it keeps trying every (s, a).

Expected Sarsa §6.6

Same loop as Q-learning; only the target changes:
 $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \sum_a \pi(a \mid S') Q(S', a) - Q(S, A) \right]$
π can be the behavior policy (on-policy) or another one (off-policy).
π greedy ⇒ exactly Q-learning.
Removes the randomness of sampling A', so it tolerates larger α than Sarsa.

Double Q-learning §6.7 · $Q1 \approx Q2 \approx q_*$

Init: Q1, Q2 arbitrary, with Q(terminal, ·) = 0
Loop for each episode: initialize S
 Loop for each step, until S is terminal:
 choose A from S, ε-greedy in Q1 + Q2; take A, observe R, S'
 with probability 0.5:
 $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha[R + \gamma Q_2(S', \arg \max_a Q_1(S', a)) - Q_1(S, A)]$
 else: the same with Q1 and Q2 swapped
 S ← S'
One table picks the action, the other scores it. Twice the memory, same work per step.

Bias, variance and batch updating

Why MC and TD behave differently with limited data (§6.2–6.3), plus the other bias–variance trade-offs in chapters 5 and 6.

Bias and variance of each target

definitions

bias = $E[\text{target}] - v_{\pi}(s)$, variance = $\text{Var}[\text{target}]$

total error

$E[(\text{target} - v_{\pi}(s))^2] = \text{bias}^2 + \text{variance}$

MC: unbiased

$E_{\pi}[G_t \mid S_t = s] = v_{\pi}(s)$

TD: biased by the error of V at the next state (zero once V = v_{π})

$E_{\pi}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s] - v_{\pi}(s) = \gamma E_{\pi}[V(S_{t+1}) - v_{\pi}(S_{t+1}) \mid S_t = s]$

MC target = TD target (in brackets) + every later TD error, from (6.6) with V held fixed

$G_t = [V(S_t) + \delta_t] + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \dots + \gamma^{T-t-1} \delta_{T-1}$

why variance grows

$\text{Var}[G_t] = \sigma^2 \sum_{k=0}^{T-t-1} \gamma^{2k}$ (independent rewards, variance σ^2 each)

MC's target carries the randomness of every remaining step. TD's carries one step, then a fixed number $V(S_{t+1})$ that already averages many past visits. DP has no sampling noise at all. TD's bias shrinks as V improves; MC's variance never does. That's why TD usually learns faster.

Exact toy: when does TD's target beat MC's?

10 steps left, $\gamma = 1$, a fixed path, each reward random with variance 1, and V wrong by e at every state. Error² of each target = bias² + variance:

target	bias	variance	e = 0.5	e = 2	e = 3.5
TD(0)	e	1	1.25	5	13.25
5-step	e	5	5.25	9	17.25
MC	0	10	10	10	10

Best target in green: TD wins while |e| < 3, i.e. once V is even roughly right.
Early in learning V can be far off, so TD starts biased; as V improves its error drops toward variance 1, while MC stays at 10. n-step methods (ch. 7) and λ slide between the two.

Other bias-variance trade-offs in chapters 5 and 6

ordinary IS (5.5)

unbiased; variance can be infinite (Example 5.5)

weighted IS (5.6)

biased, bias $\rightarrow 0$; variance $\rightarrow 0$ with more data (bounded returns)

per-decision IS

same expectation as ordinary IS, often lower variance

Expected Sarsa

same expectation as Sarsa, minus the noise of sampling A_{t+1}

Q-learning

$E[\max_a Q] \geq \max_a q$: positive (maximization) bias

Double Q-learning

unbiased for the chosen action; can slightly underestimate

constant α vs 1/n

constant α : tracks change, never stops fluctuating; 1/n: converges

Batch updating

\$6.3

Given: a fixed set of episodes; a small step size α
Repeat (one sweep over the whole dataset):

- hold V fixed, and compute the increment for every time step t:
 - MC: $\alpha[G_t - V(S_t)]$ TD(0): $\alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- for each state s, add up the increments from all of its visits
- apply each state's sum once

until V stops changing

The answer depends only on the data: not on α (if small enough) and not on the initial V. Holding V fixed within a sweep is the same idea as frozen targets and target networks.

Where batch MC converges

one sweep

$V_{k+1}(s) = V_k(s) + \alpha \sum_{t: S_t = s} (G_t - V_k(s))$

stops when

$\sum_{t: S_t = s} (G_t - V(s)) = 0 \Rightarrow V(s) = \frac{1}{N(s)} \sum_{t: S_t = s} G_t = \bar{G}(s)$

equivalently

$V = \arg \min_V \sum_t (G_t - V(S_t))^2$ (least squares on the data)

error per sweep

$V_{k+1}(s) - \bar{G}(s) = (1 - \alpha N(s))(V_k(s) - \bar{G}(s))$

Converges when $0 < \alpha N(s) < 2$ (smoothly when $\alpha N(s) < 1$): that's what “ α small enough” means.

Where batch TD(0) converges

one sweep

$V_{k+1}(s) = V_k(s) + \alpha \sum_{t: S_t = s} (R_{t+1} + \gamma V_k(S_{t+1}) - V_k(s))$

stops when

$\sum_{t: S_t = s} (R_{t+1} + \gamma V(S_{t+1}) - V(s)) = 0$

group by next state

$\frac{1}{N(s)} \sum_{t: S_t = s} V(S_{t+1}) = \sum_{s'} \hat{P}(s' \mid s) V(s'), \quad \hat{P}(s' \mid s) = \frac{N(s \rightarrow s')}{N(s)}$

result

$V(s) = \hat{r}(s) + \gamma \sum_{s'} \hat{P}(s' \mid s) V(s'), \quad \hat{r}(s) = \text{mean reward leaving } s$

This is the Bellman equation of the maximum-likelihood Markov model of the data: the certainty-equivalence estimate. Solving it directly takes $\sim |S|^2$ memory and $\sim |S|^3$ steps; TD gets there with $|S|$ memory by sweeping.

Worked example: 8 episodes, $\alpha = 0.05$, starting from V = (5, −3)

Data: A, 0, B, 0 once; then B alone seven times: four with reward 1, three with 0 ($\gamma = 1$).

sweeps	MC: V(A)	MC: V(B)	TD: V(A)	TD: V(B)
1	4.75	-1.60	4.60	-1.60
5	3.87	0.23	3.63	0.23
20	1.79	0.50	1.93	0.50
100	0.03	0.50	0.52	0.50
400	0.00	0.50	0.50	0.50

MC ends at V(A) = 0 (its one observed return) and TD at V(A) = 0.5 (= V(B)).
MC's V(A) creeps (factor 0.95 per sweep: one visit), V(B) is fast (factor 0.6: eight visits). Start from (0, 0) instead and both land on exactly the same values.

Batch vs. normal (online) updating

Normal updating moves V only part of the way toward these batch answers after each step or episode. Batch TD beats batch MC on the random walk (Example 6.3): its answer uses the Markov structure. MC's answer can be better when the states aren't truly Markov, since it assumes nothing about what follows. Batch MC fits the returns it has seen; batch TD fits a model first, then computes that model's values.