

# Model Predictive Control and Reinforcement Learning

## – Lecture 04: Monte Carlo and Temporal-Difference Learning –

Jasper Hoffmann

with material adapted from Joschka Boedecker and Gabriel Kalweit

University of Freiburg

School on Model Predictive Control and Reinforcement Learning  
Freiburg, 7-16 October 2026

universität freiburg



NTNU

Norwegian University of  
Science and Technology

Last lecture: planning by dynamic programming, solve a *known, discrete* MDP.

## Policy Iteration

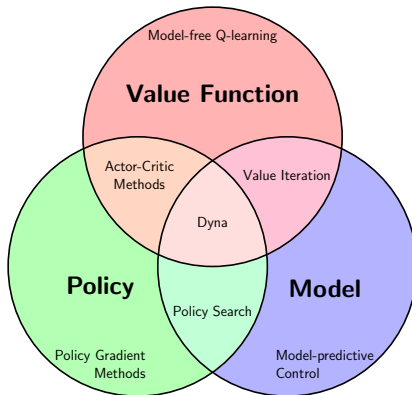
Alternate **evaluating** the value function  $V^\pi$  and **improving** the policy  $\pi$  to convergence.

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi^\star \xrightarrow{E} V^\star$$

## Value Iteration

Evaluate just once and combine it with the policy improvement step, iterating  $j = 0, 1, 2, \dots$  until convergence:

$$V_{j+1}(s) = \min_a \left\{ \ell(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V_j(s') \right\}$$



**Last lecture:** value functions and policies with access to a *given* model.

**Today:** value functions and policies learned from samples, *without* a model (model-free).



Dynamic programming is optimal, but it requires knowledge of the dynamics and is too computationally expensive → approximate dynamic programming:

$$V_{j+1}(s) = \min_a \left\{ \ell(s, a) + \gamma \sum_{s'} P(s' | s, a) V_j(s') \right\}$$

- ▶ **Local state updates:** update only the visited states  $s, s'$  instead of sweeping over all states.
- ▶ **No model required:** the transition model  $P$  and the cost  $\ell$  are not needed; we sample  $s'$  and the cost directly from interaction. *(this lecture)*
- ▶ **Function approximation:** replace the table  $V$  by a function approximator. *(next lecture)*

Monte Carlo Prediction

Monte Carlo Control

Temporal-Difference Prediction

Temporal-Difference Control

Summary

**Monte Carlo (MC)** methods estimate expectations by averaging random samples:

$$\mathbb{E}[f(X)] = \int f(x) p(x) dx \approx \frac{1}{N} \sum_{i=1}^N f(X_i), \quad X_1, \dots, X_N \stackrel{\text{i.i.d.}}{\sim} p.$$

If  $\sigma^2 = \text{Var}[f(X)] < \infty$ , the error shrinks like  $\sigma/\sqrt{N}$ , at a rate independent of the dimension of  $X$  (proof in the appendix).

In RL: model-free prediction and control, i.e. estimate and optimize the value function of an *unknown* MDP  $\mathcal{M}$ .

- ▶ MC methods learn from sampled episodes  $S_0, A_0, S_1, A_1, \dots, S_T$  of *experience*
- ▶ MC is model-free: no knowledge of the transition model  $P$  is required
- ▶ MC averages the returns of complete episodes, without bootstrapping (see TD)
- ▶ Here we assume episodes terminate almost surely

**Goal:** learn the state-value function  $V^\pi$  for a given policy  $\pi$ .

**Idea:** average the returns observed from each state.

**Recall:** the value function is the expected *return*, the discounted cumulative cost:

$$V^\pi(s) = \mathbb{E}_\pi[G_k \mid S_k = s],$$
$$G_k = \ell(S_k, A_k) + \gamma \ell(S_{k+1}, A_{k+1}) + \dots + \gamma^{T-k-1} \ell(S_{T-1}, A_{T-1}).$$

$T$ : random termination time;  $S_T$  is terminal with zero terminal cost.

**Monte Carlo:** replace the expectation by the average of the  $N(s)$  returns observed from  $s$ :

$$V^\pi(s) \approx \frac{1}{N(s)} \sum_{i=1}^{N(s)} G^{(i)}(s),$$

where  $G^{(i)}(s)$  is the  $i$ -th return observed from  $s$ .

- ▶ First-visit MC: estimates  $V^\pi(s)$  as the average of the returns following first visits to  $s$ .
- ▶ Every-visit MC: estimates  $V^\pi(s)$  as the average of the returns following all visits to  $s$ .

## First-visit MC prediction, for estimating $V \approx V^\pi$

Input: a policy  $\pi$  to be evaluated

Initialize:

$V(s) \in \mathbb{R}$ , arbitrarily, for all  $s \in \mathcal{S}$

$Returns(s) \leftarrow$  an empty list, for all  $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following  $\pi$ :  $S_0, A_0, S_1, A_1, \dots, S_{T-1}, A_{T-1}, S_T$

$G \leftarrow 0$

Loop for each step of episode,  $k = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \ell(S_k, A_k) + \gamma G$

Unless  $S_k$  appears in  $S_0, S_1, \dots, S_{k-1}$ :

Append  $G$  to  $Returns(S_k)$

$V(S_k) \leftarrow \text{average}(Returns(S_k))$

**Game:** a card sum close to 21 without exceeding it, against a dealer (infinite deck).

- ▶ **States** (200): current sum (12–21), dealer's showing card (ace–10), usable ace (yes/no)
- ▶ **Actions:** *stick* (stop and terminate) or *hit* (take another card)
- ▶ **Rewards:** +1 win, 0 draw, −1 loss; going over 21 loses immediately; 0 otherwise
- ▶ **Evaluated policy** (next slide): stick only on 20 or 21

## Notes:

- ▶ An ace counts as 1 or 11; it is *usable* if it can count as 11 without exceeding 21.
- ▶ Cards are dealt automatically until the sum is at least 12, since hitting below 12 is safe.

Here and in the next plot: reward maximization; equivalently, cost = negative reward.

# Example: Blackjack

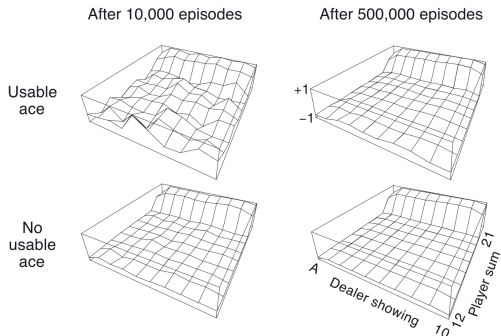


**Figure 5.1:** Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation. ■

MC needs many samples: each state is estimated independently from its own returns, so rarely visited states (usable ace) converge slowest.

Monte Carlo Prediction

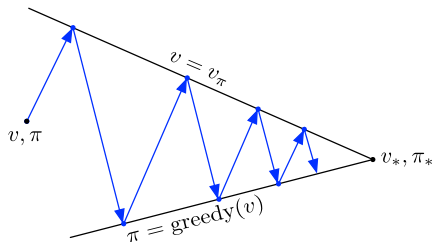
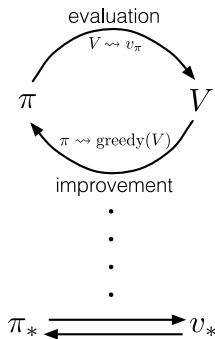
Monte Carlo Control

Temporal-Difference Prediction

Temporal-Difference Control

Summary

# Generalized Policy Iteration with Monte Carlo



- Monte Carlo prediction (policy evaluation):  $V \approx V^\pi$
- Policy improvement: greedy?



**Problem:** given a value estimate  $V$ , choosing the best action requires a model of the MDP:

$$\pi(s) \in \arg \min_{a \in \mathcal{A}} \left\{ \ell(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V(s') \right\}$$

**Idea:** estimate action values  $Q(s, a)$  instead; then the best action needs no model:

$$\pi(s) \in \arg \min_{a \in \mathcal{A}} Q(s, a)$$

**Monte Carlo:** average the returns observed after taking  $a$  in  $s$ :

$$Q^\pi(s, a) \approx \frac{1}{N(s, a)} \sum_{i=1}^{N(s, a)} G^{(i)}(s, a)$$

**Generalized policy iteration:** alternate MC prediction of  $Q$  and policy improvement. Greedy?

A **multi-armed bandit** is an MDP with a single state: in each round, pick an arm  $a \in \mathcal{A}$  and observe the cost  $\ell(a, W)$ , where  $W \sim p$  is unknown random noise. The value of an arm is

$$Q(a) = \mathbb{E}[\ell(a, W)] \approx \frac{1}{N(a)} \sum_{i=1}^{N(a)} \ell(a, W_i).$$

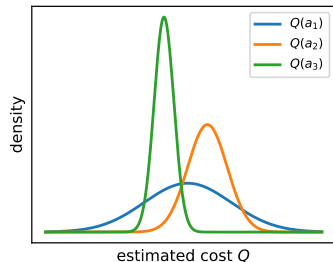


*The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. [...] On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward.*

Sutton & Barto, 2018

**Example:** going out for dinner, do you go to your favourite restaurant or try a new one? And if a new one was bad once, was it the restaurant or bad luck?

- ▶  **$\epsilon$ -greedy:** with probability  $1 - \epsilon$  the greedy arm, with probability  $\epsilon$  a random arm; often with a decaying  $\epsilon$
- ▶ **Softmax (Boltzmann):**  $\pi(a) \propto \exp(-Q(a)/\tau)$  with temperature  $\tau$
- ▶ **Optimistic initial values:** start with low cost estimates, so every arm gets tried
- ▶ **Upper confidence bound (UCB):**  
 $\arg \min_a \{Q(a) - c\sqrt{\ln n / N(a)}\}$ , i.e. prefer arms that are still uncertain



Which arm should we pick?

Similar strategies can be applied to  $Q(s, a)$  in MDPs; here we use  $\epsilon$ -greedy.

Bandit theory: Bubeck & Cesa-Bianchi (2012).

## On-policy first-visit MC control (for $\epsilon$ -soft policies), estimates $\pi \approx \pi^*$

Algorithm parameter: small  $\epsilon > 0$

Initialize:

$\pi \leftarrow$  an arbitrary  $\epsilon$ -soft policy

$Q(s, a) \in \mathbb{R}$  (arbitrarily), for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$  empty list, for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following  $\pi$ :  $S_0, A_0, S_1, A_1, \dots, S_{T-1}, A_{T-1}, S_T$

$G \leftarrow 0$

Loop for each step of episode,  $k = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \ell(S_k, A_k) + \gamma G$

Unless the pair  $S_k, A_k$  appears in  $S_0, A_0, S_1, A_1, \dots, S_{k-1}, A_{k-1}$ :

Append  $G$  to  $Returns(S_k, A_k)$

$Q(S_k, A_k) \leftarrow \text{average}(Returns(S_k, A_k))$

$A^* \leftarrow \arg \min_a Q(S_k, a)$

(with ties broken arbitrarily)

For all  $a \in \mathcal{A}(S_k)$ :  $\pi(a | S_k) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(S_k)| & \text{if } a = A^* \\ \epsilon/|\mathcal{A}(S_k)| & \text{if } a \neq A^* \end{cases}$

$\epsilon$ -soft:  $\pi(a | s) \geq \epsilon/|\mathcal{A}(s)|$  for all  $s, a$ .  $\epsilon$ -greedy w.r.t.  $Q^\pi$  is at least as good as  $\pi$  (appendix); softmax also explores.



**Problem:** on-policy MC has to keep exploring, so it only learns the best  $\epsilon$ -soft policy, not the optimal one.

**Idea:** use two policies.

- ▶ **Target policy**  $\pi$ : the policy whose values the update estimates, e.g. greedy
- ▶ **Behaviour policy**  $b$ : the policy that generates the data, e.g.  $\epsilon$ -greedy

|                                                                 |
|-----------------------------------------------------------------|
| <b>Off-policy:</b> $b \neq \pi$ . <b>On-policy:</b> $b = \pi$ . |
|-----------------------------------------------------------------|



**Problem:** returns generated by  $b$  estimate the wrong values,

$$\mathbb{E}_b[G_k \mid S_k = s] = V^b(s), \quad \text{but we want} \quad V^\pi(s) = \mathbb{E}_\pi[G_k \mid S_k = s].$$

**Probability of a trajectory** under a policy  $\pi$ , starting in  $S_k$ :

$$\Pr_\pi(A_k, S_{k+1}, \dots, S_T \mid S_k) = \prod_{i=k}^{T-1} \pi(A_i \mid S_i) P(S_{i+1} \mid S_i, A_i).$$

**Idea:** weight each return by how much more likely its trajectory is under  $\pi$  than under  $b$ .

**Importance sampling ratio:** the relative probability of the trajectory,

$$\begin{aligned}\rho_{k:T-1} &= \frac{\Pr_{\pi}(A_k, S_{k+1}, \dots, S_T \mid S_k)}{\Pr_b(A_k, S_{k+1}, \dots, S_T \mid S_k)} \\ &= \frac{\prod_{i=k}^{T-1} \pi(A_i \mid S_i) P(S_{i+1} \mid S_i, A_i)}{\prod_{i=k}^{T-1} b(A_i \mid S_i) P(S_{i+1} \mid S_i, A_i)} = \prod_{i=k}^{T-1} \frac{\pi(A_i \mid S_i)}{b(A_i \mid S_i)}\end{aligned}$$

- ▶ The transition model  $P$  cancels, so no model is needed.
- ▶ Requires *coverage*,  $\pi(a \mid s) > 0 \Rightarrow b(a \mid s) > 0$ , so that  $\rho$  is defined.
- ▶ Weighted returns have the right expectation:  $\mathbb{E}_b[\rho_{k:T-1} G_k \mid S_k = s] = V^{\pi}(s)$ .
- ▶ But a product of many ratios can have a very high variance.

## Off-policy MC control, for estimating $\pi \approx \pi^*$

Initialize, for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$ :

$Q(s, a) \in \mathbb{R}$  (arbitrarily)

$C(s, a) \leftarrow 0$

$\pi(s) \leftarrow \arg \min_a Q(s, a)$  (with ties broken consistently)

Loop forever (for each episode):

$b \leftarrow$  any soft policy

Generate an episode using  $b$ :  $S_0, A_0, S_1, A_1, \dots, S_{T-1}, A_{T-1}, S_T$

$G \leftarrow 0$ ,  $W \leftarrow 1$

Loop for each step of episode,  $k = T-1, T-2, \dots, 0$ :

$G \leftarrow \ell(S_k, A_k) + \gamma G$

$C(S_k, A_k) \leftarrow C(S_k, A_k) + W$

$Q(S_k, A_k) \leftarrow Q(S_k, A_k) + \frac{W}{C(S_k, A_k)} [G - Q(S_k, A_k)]$

$\pi(S_k) \leftarrow \arg \min_a Q(S_k, a)$  (with ties broken consistently)

If  $A_k \neq \pi(S_k)$  then exit inner loop (proceed to next episode)

$W \leftarrow W \frac{1}{b(A_k|S_k)}$

$C$ : total weight.  $W$ : IS ratio (numerator 1 for greedy  $\pi$ ). Only returns after the last non-greedy action contribute: long horizons  $\Rightarrow$  rare samples, large weights  $\Rightarrow$  inefficient  $\rightarrow$  TD.

Monte Carlo Prediction

Monte Carlo Control

Temporal-Difference Prediction

Temporal-Difference Control

Summary



Estimate and optimize the value function of an *unknown* MDP using temporal-difference learning.

- ▶ TD combines Monte Carlo and dynamic programming ideas
- ▶ Like MC, TD learns directly from raw experience without a model of the dynamics
- ▶ TD learns from *incomplete* episodes by bootstrapping
- ▶ Bootstrapping: update an estimate based on other estimates, without waiting for a final outcome (update a guess towards a guess)

## Monte Carlo update

Update value  $V(S_k)$  towards the *actual* return  $G_k$ :

$$V(S_k) \leftarrow V(S_k) + \alpha [G_k - V(S_k)]$$

## Simplest temporal-difference learning algorithm: TD(0)

Update value  $V(S_k)$  towards the *estimated* return  $\ell(S_k, A_k) + \gamma V(S_{k+1})$ :

$$V(S_k) \leftarrow V(S_k) + \alpha [\ell(S_k, A_k) + \gamma V(S_{k+1}) - V(S_k)]$$

- ▶  $\ell(S_k, A_k) + \gamma V(S_{k+1})$  is called the *TD target*
- ▶  $\delta_k = \ell(S_k, A_k) + \gamma V(S_{k+1}) - V(S_k)$  is called the *TD error*



## Policy evaluation with a model: Bellman expectation operator

$$(\mathcal{T}^\pi V)(s) = \sum_a \pi(a | s) \left[ \ell(s, a) + \gamma \sum_{s'} P(s' | s, a) V(s') \right]$$

Dynamic programming:  $V(s) \leftarrow (\mathcal{T}^\pi V)(s)$ ; fixed point  $V^\pi = \mathcal{T}^\pi V^\pi$ .

## Without a model: sample one transition

For  $A \sim \pi(\cdot | s)$  and  $S' \sim P(\cdot | s, A)$ , use the sample target

$$\hat{\mathcal{T}}^\pi V(s) = \ell(s, A) + \gamma V(S'), \quad \mathbb{E}[\hat{\mathcal{T}}^\pi V(s)] = (\mathcal{T}^\pi V)(s).$$

For a fixed current  $V$ : unbiased for its Bellman update, not necessarily for  $V^\pi(s)$ .

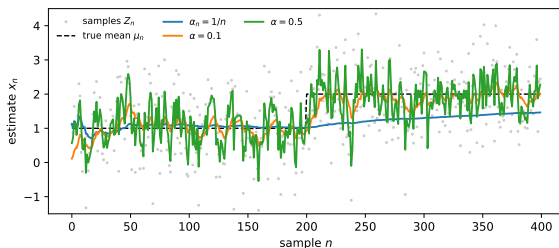
**TD(0)**: take a step towards this noisy target rather than replacing  $V(s)$ :

$$V(s) \leftarrow V(s) + \alpha \underbrace{[\ell(s, A) + \gamma V(S') - V(s)]}_{\text{sampled Bellman residual / TD error}}$$

**Theory note:** how should we choose the step size?

$$x_{n+1} = x_n + \alpha_n (Z_n - x_n),$$

For i.i.d. samples with a fixed mean and finite variance,  $x_n \rightarrow \mathbb{E}[Z]$  almost surely under the **Robbins–Monro conditions**:  $\sum_n \alpha_n = \infty$  and  $\sum_n \alpha_n^2 < \infty$  (e.g.  $\alpha_n = 1/n$ ).



**In practice:** targets change as the policy and bootstrap estimates change. Too much step-size decay slows tracking; a constant  $\alpha$  remains responsive but noisy.

## Tabular TD(0) for estimating $V^\pi$

Input: the policy  $\pi$  to be evaluated

Algorithm parameter: step size  $\alpha \in (0, 1]$

Initialize  $V(s)$ , for all  $s \in \mathcal{S}^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop for each episode:

    Initialize  $S$

    Loop for each step of episode:

$A \leftarrow$  action given by  $\pi$  for  $S$

        Take action  $A$ , observe the cost  $\ell(S, A)$  and  $S'$

$V(S) \leftarrow V(S) + \alpha [\ell(S, A) + \gamma V(S') - V(S)]$

$S \leftarrow S'$

    until  $S$  is terminal

For a random variable  $Z$  that estimates a quantity  $\theta$ :

$$\mathbf{bias} = \mathbb{E}[Z] - \theta, \quad \mathbf{variance} = \mathbb{E}[(Z - \mathbb{E}[Z])^2].$$

Here,  $Z$  is the target and  $\theta = V^\pi(S_k)$ .

**MC target**  $G_k = \ell(S_k, A_k) + \gamma \ell(S_{k+1}, A_{k+1}) + \dots$  sums all remaining costs: unbiased, but random over the whole rest of the episode, so **typically higher variance**.

**TD target**  $\ell(S_k, A_k) + \gamma V(S_{k+1})$  is random only in one step. The rest of the episode is replaced by the estimate  $V$  (bootstrapping), so **typically lower variance**, but **potentially biased** while  $V \neq V^\pi$ .

In addition, TD learns after every step and does not need the episode to end.

Monte Carlo Prediction

Monte Carlo Control

Temporal-Difference Prediction

**Temporal-Difference Control**

Summary

Same idea as MC control (generalized policy iteration): estimate  $Q$  and act  $\epsilon$ -greedily. But update  $Q$  after every step with TD, instead of waiting for the return of the episode.

The two methods differ only in the bootstrap target:

|                                        |                                                       |
|----------------------------------------|-------------------------------------------------------|
| SARSA: $\ell(S, A) + \gamma Q(S', A')$ | Q-learning: $\ell(S, A) + \gamma \min_{a'} Q(S', a')$ |
|----------------------------------------|-------------------------------------------------------|

SARSA uses the next action  $A'$  that is actually taken, Q-learning the best next action.

- ▶ SARSA: each update uses  $(S, A, \ell(S, A), S', A')$ , i.e. state, action, cost, state, action
- ▶ Why is it considered an on-policy method?

## Sarsa (on-policy TD control) for estimating $Q \approx Q^*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\epsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+$ ,  $a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

    Loop for each step of episode:

        Take action  $A$ , observe the cost  $\ell(S, A)$  and  $S'$

        Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

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

$S \leftarrow S'; A \leftarrow A'$

    until  $S$  is terminal

Tabular SARSA converges to  $Q^*$  with suitable diminishing step sizes and  $\epsilon \rightarrow 0$ , while every state-action pair is visited infinitely often.

- Why is it considered an off-policy method?

## Q-learning (off-policy TD control) for estimating $\pi \approx \pi^*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\epsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+$ ,  $a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

    Loop for each step of episode:

        Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

        Take action  $A$ , observe the cost  $\ell(S, A)$  and  $S'$

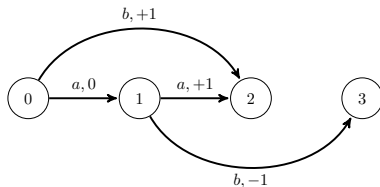
$Q(S, A) \leftarrow Q(S, A) + \alpha [\ell(S, A) + \gamma \min_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

    until  $S$  is terminal

Q-learning (Watkins, 1989) converges to  $Q^*$  for tabular  $Q$  if all state-action pairs keep being visited and the step sizes decrease suitably (Watkins & Dayan, 1992). It is the basis of DQN (next lecture).

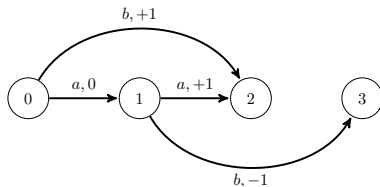
## Example: SARSA vs Q-learning Updates



Episodes (edges: action, cost):  $0 \xrightarrow{a} 1 \xrightarrow{a} 2$ ,  $0 \xrightarrow{a} 1 \xrightarrow{b} 3$ ,  $0 \xrightarrow{a} 1 \xrightarrow{a} 2$ .

**Question:** with  $Q = 0$ ,  $\alpha = 1$ ,  $\gamma = 1$ , what is  $Q(0, a)$  after the three episodes with SARSA and with Q-learning?

## Example: SARSA vs Q-learning Updates



Episodes (edges: action, cost):  $0 \xrightarrow{a} 1 \xrightarrow{a} 2$ ,  $0 \xrightarrow{a} 1 \xrightarrow{b} 3$ ,  $0 \xrightarrow{a} 1 \xrightarrow{a} 2$ .

**Question:** with  $Q = 0$ ,  $\alpha = 1$ ,  $\gamma = 1$ , what is  $Q(0, a)$  after the three episodes with SARSA and with Q-learning?

|                           | episode 1 | episode 2 | episode 3 |
|---------------------------|-----------|-----------|-----------|
| $Q(1, a)$                 | 1         | 1         | 1         |
| $Q(1, b)$                 | 0         | -1        | -1        |
| $Q(0, a)$ with SARSA      | 0         | 0         | 1         |
| $Q(0, a)$ with Q-learning | 0         | 0         | -1        |

In episode 3, SARSA uses the action actually taken in state 1 ( $a$ ), Q-learning the best one ( $b$ ).

Monte Carlo Prediction

Monte Carlo Control

Temporal-Difference Prediction

Temporal-Difference Control

Summary

- ▶ **Approximate dynamic programming:** learn from samples without a model, updating only visited states.
- ▶ **MC:** average complete returns, without bootstrapping; typically higher variance.
- ▶ **TD:** use sampled Bellman updates with bootstrapping; typically lower variance, but biased targets while learning.
- ▶ **Control** alternates learning  $Q$  and improving the policy while maintaining exploration.
- ▶ **SARSA** learns on-policy; **Q-learning** uses an off-policy greedy target. Off-policy MC uses importance sampling.
- ▶ **Step sizes** balance convergence and tracking: decreasing  $\alpha$  reduces noise, but can slow adaptation to changing targets.

**Next: replace value tables by function approximators.**



- ▶ Richard S. Sutton & Andrew G. Barto: *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.  
Chapters 5–6: Monte Carlo and temporal-difference learning.  
<http://incompleteideas.net/book/the-book.html>
- ▶ Sébastien Bubeck & Nicolò Cesa-Bianchi: *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Foundations and Trends in Machine Learning, 2012.  
<https://arxiv.org/abs/1204.5721>

For  $0 < \epsilon < 1$ , let  $\pi$  be  $\epsilon$ -soft and  $\pi'$  be  $\epsilon$ -greedy with respect to  $Q^\pi$ . Fix  $s$  and write  $m = |\mathcal{A}(s)|$ . Then

$$\begin{aligned}(\mathcal{T}^{\pi'} V^\pi)(s) &= \sum_a \pi'(a | s) Q^\pi(s, a) \\&= \frac{\epsilon}{m} \sum_a Q^\pi(s, a) + (1 - \epsilon) \min_a Q^\pi(s, a) \\&\leq \frac{\epsilon}{m} \sum_a Q^\pi(s, a) + (1 - \epsilon) \sum_a \frac{\pi(a | s) - \frac{\epsilon}{m}}{1 - \epsilon} Q^\pi(s, a) \\&= \sum_a \pi(a | s) Q^\pi(s, a) = V^\pi(s)\end{aligned}$$

The inequality holds because the weights  $\frac{\pi(a|s) - \epsilon/m}{1 - \epsilon}$  are non-negative (as  $\pi$  is  $\epsilon$ -soft) and sum to 1, and a weighted average is never below the minimum. By the policy improvement theorem,  $V^{\pi'}(s) \leq V^\pi(s)$  for all  $s$ .

For  $\epsilon = 1$ , both policies are uniform and their values are equal.

Let  $X_1, \dots, X_N \stackrel{\text{i.i.d.}}{\sim} p$ ,  $\mu = \mathbb{E}[f(X)]$ ,  $\sigma^2 = \text{Var}[f(X)] < \infty$ , and  $\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^N f(X_i)$ .

- ▶ **Unbiased:** by linearity of the expectation,  $\mathbb{E}[\hat{\mu}_N] = \frac{1}{N} \sum_{i=1}^N \mathbb{E}[f(X_i)] = \mu$ .
- ▶ **Variance:** the samples are independent, so their variances add,

$$\text{Var}[\hat{\mu}_N] = \frac{1}{N^2} \sum_{i=1}^N \text{Var}[f(X_i)] = \frac{\sigma^2}{N}.$$

- ▶ Hence the root-mean-square error is  $\sqrt{\mathbb{E}[(\hat{\mu}_N - \mu)^2]} = \sqrt{\text{Var}[\hat{\mu}_N]} = \sigma/\sqrt{N}$ ; the dimension of  $X$  enters only through  $\sigma$ .

This variance formula assumes independent samples; every-visit returns within an episode can be correlated.

**Note.** See A. B. Owen, *Monte Carlo theory, methods and examples*, Ch. 2, <https://artowen.su.domains/mc/>.