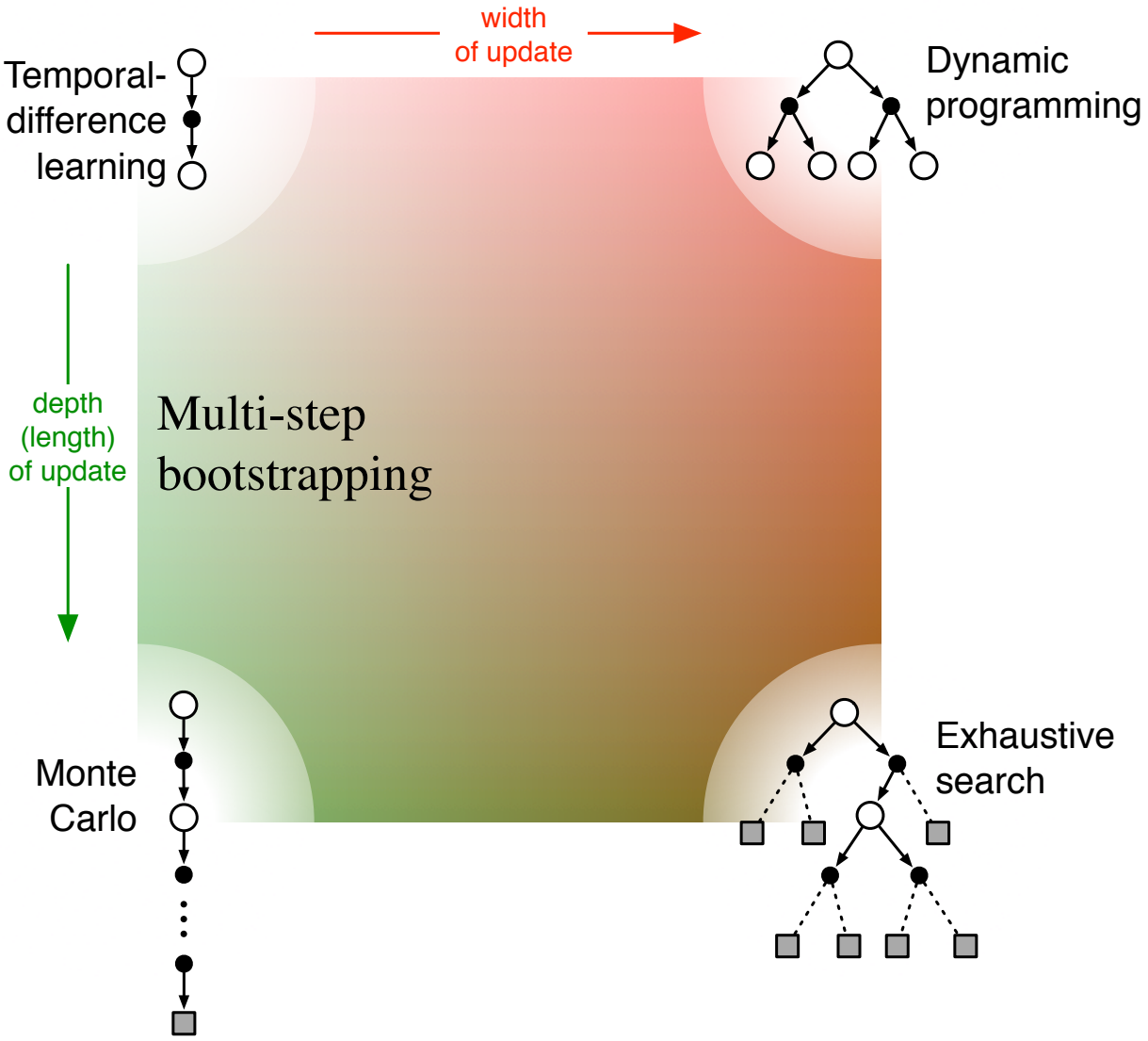
RL: Up to this Point

- Policy evaluation: backups without a max (prediction)
- Policy improvement: form a greedy policy, if only locally
- Policy iteration: alternate the above two processes (control)
- Value iteration: backups with a max (control)

- Full backups (to be contrasted later with sample backups)
- Generalized Policy Iteration (GPI)
- Asynchronous DP: a way to avoid exhaustive sweeps
- **Bootstrapping**: updating estimates based on other estimates
- Biggest limitation of DP is that it requires a *probability model* (as opposed to a generative or simulation model)

Unified View

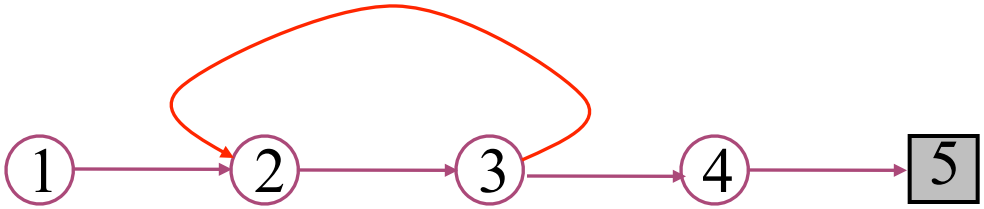


Monte Carlo Methods

- Monte Carlo methods are learning methods
  Experience → values, policy
- Monte Carlo methods can be used in two ways:
  - *model-free:* No model necessary and still attains optimality
  - *simulated:* Needs only a simulation, not a *full* model
- Monte Carlo methods learn from *complete* sample returns
  - Only defined for episodic tasks (for now)
- Like an associative version of a bandit method
- Can define method that is
  - *on-policy:* performing value backups for the sampled action
  - *off-policy:* performing value backups of some other action

Monte Carlo Policy Evaluation

- *Goal:* learn $v_\pi(s)$
- *Given:* some number of episodes under π which contain *s*
- *Idea:* Average returns observed after visits to s



- *Every-Visit MC:* average returns for *every* time *s* is visited in an episode
- *First-visit MC:* average returns only for *first* time *s* is visited in an episode
- Both converge asymptotically

First-visit Monte Carlo policy evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

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

Repeat forever:

Generate an episode using π

For each state s appearing in the episode:

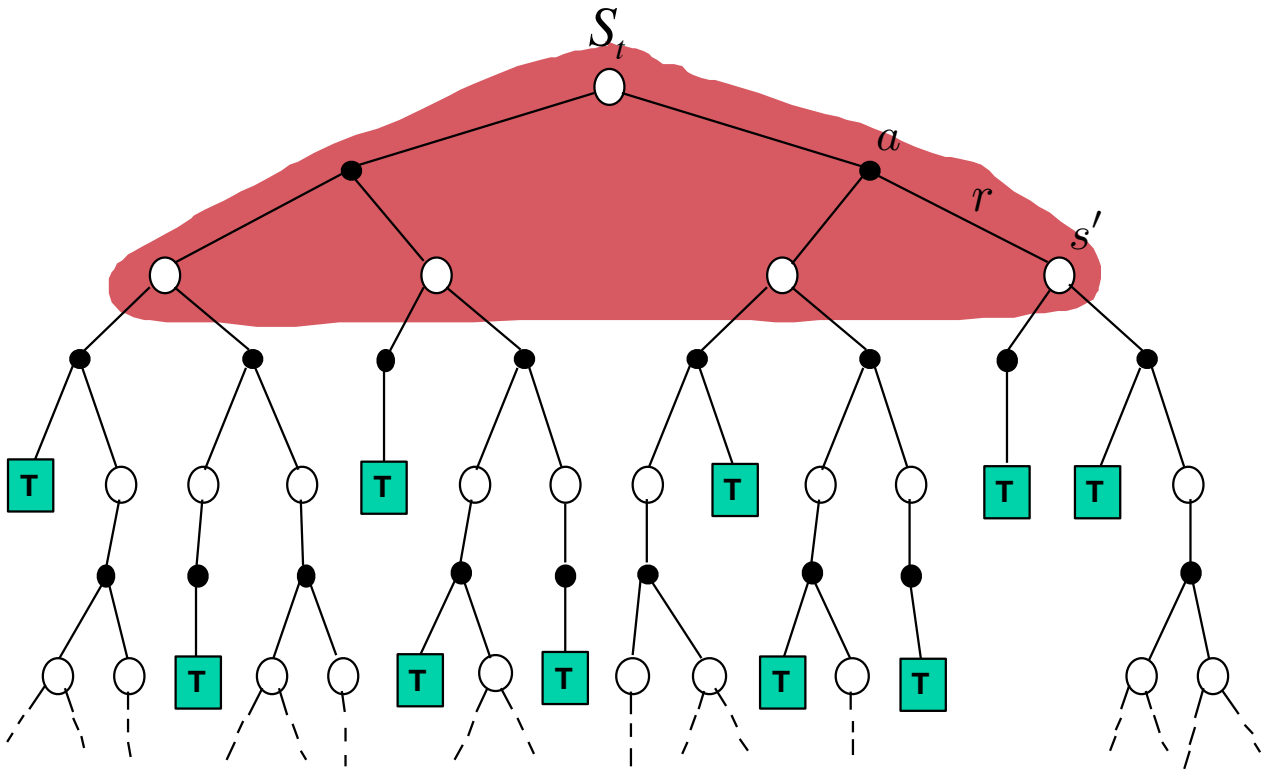
$G \leftarrow$ return following the first occurrence of s

Append G to $Returns(s)$

$V(s) \leftarrow$ average$(Returns(s))$

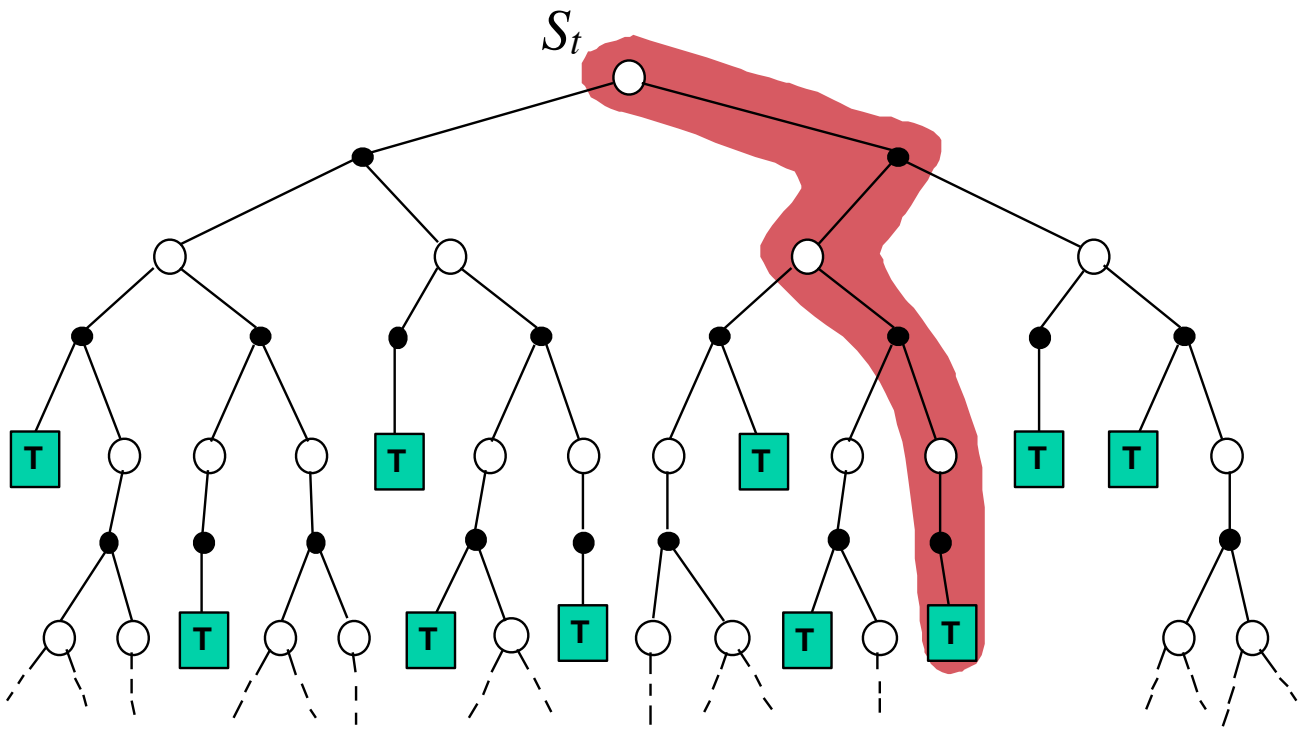
Dynamic programming

$$V(S_t) \leftarrow E_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right] = \sum_a \pi(a|S_t) \sum_{s',r} p(s', r|S_t, a)[r + \gamma V(s')]$$



Simple Monte Carlo

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

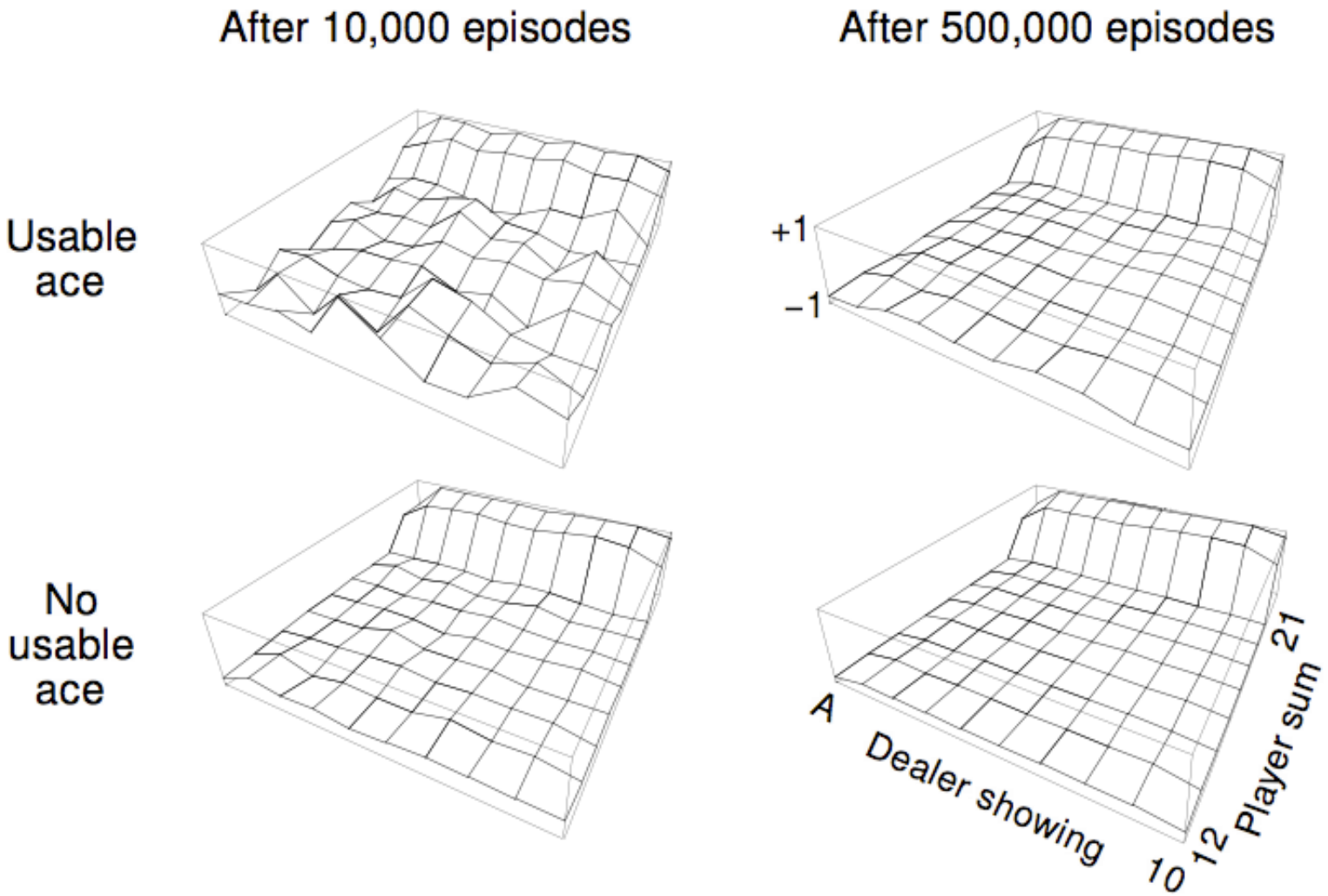


Blackjack example

- *Object:* Have your card sum be greater than the dealer's without exceeding 21. (Face cards=10; Ace=1 or 11)
- *States* (200 of them):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace?

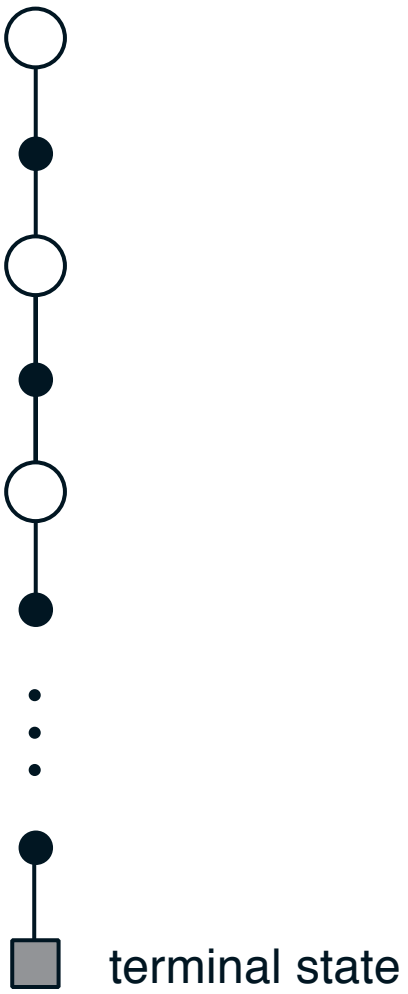
- *Reward:* +1 for winning, 0 for a draw, -1 for losing
- *Actions:* stick (stop receiving cards), hit (receive another card)
- *Policy:* Stick if my sum is 20 or 21, else hit
- No discounting ($\gamma = 1$)

Learned blackjack state-value functions



Backup diagram for Monte Carlo

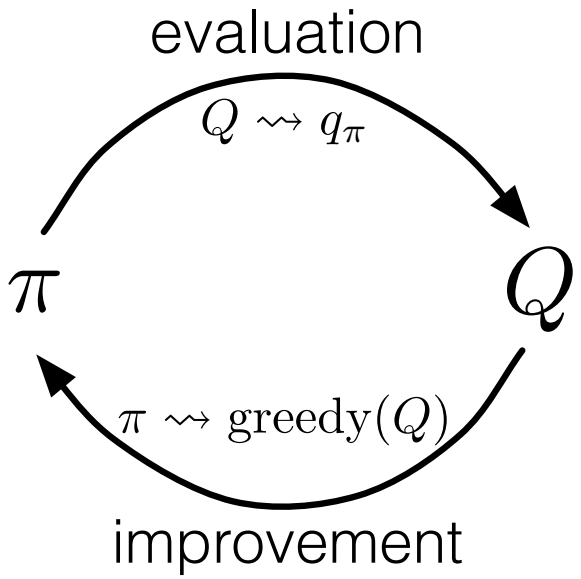
- Entire rest of episode included
- Only one choice considered at each state (unlike DP)
  - thus, there will be an explore/exploit dilemma
- Does not bootstrap from successor states's values (unlike DP)
- Time required to estimate one state does not depend on the total number of states



Monte Carlo Estimation of Action Values (Q)

- Monte Carlo is most useful when a **model is not available**
  - We want to learn q_*
- $q_\pi(s,a)$ - average return starting from state s and action a following π
- Converges asymptotically *if* every state-action pair is visited
- *Exploring starts:* Every state-action pair has a non-zero probability of being the starting pair

Monte Carlo Control



- MC policy iteration: Policy evaluation using MC methods followed by policy improvement
- Policy improvement step: greedify with respect to value (or action-value) function

Convergence of MC Control

- Greedified policy meets the conditions for policy improvement:

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$

- And thus must be $\geq \pi_k$ by the policy improvement theorem
- This assumes exploring starts and infinite number of episodes for MC policy evaluation
- To solve the latter:
  - update only to a given level of performance
  - alternate between evaluation and improvement per episode

Monte Carlo Exploring Starts

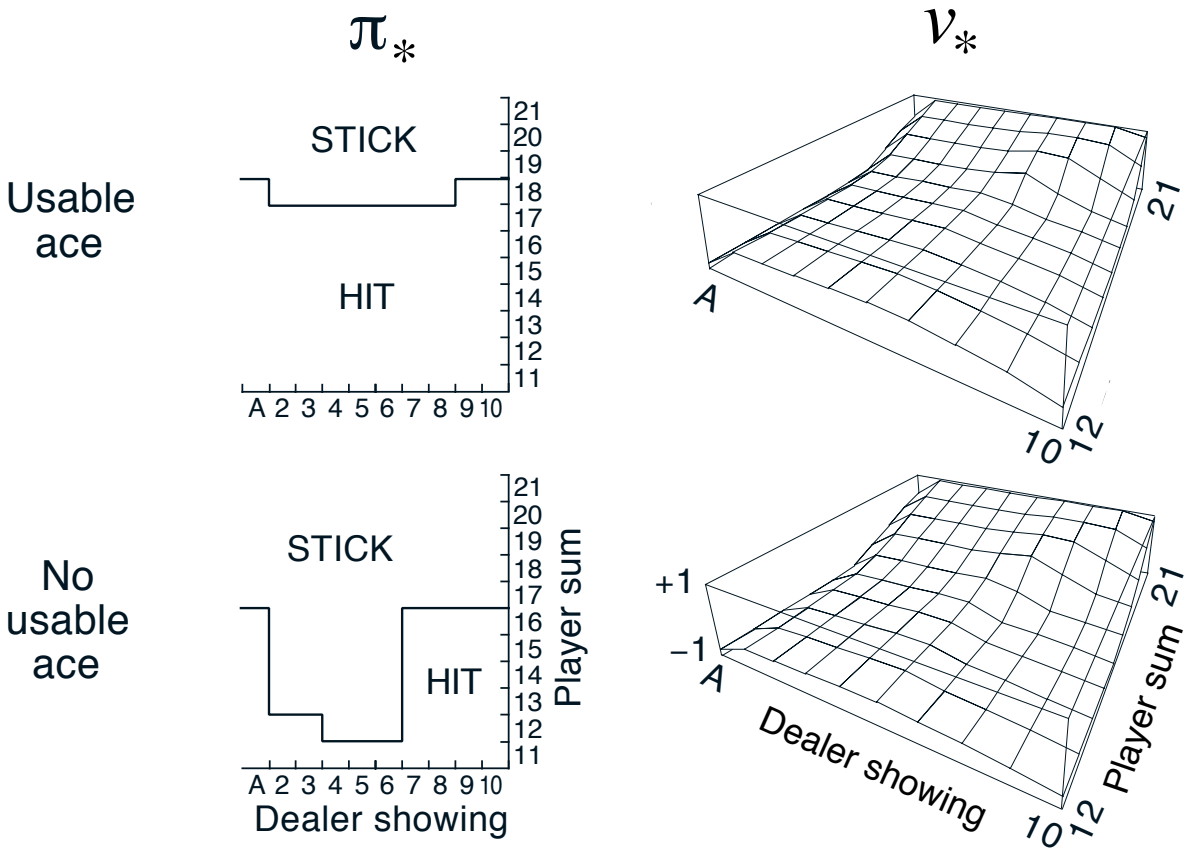
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list

Fixed point is optimal policy π^*

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
Generate an episode starting from S_0, A_0, following π
For each pair s, a appearing in the episode:
$G \leftarrow$ return following the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Blackjack example continued

- Exploring starts
- Initial policy as described before



On-policy Monte Carlo Control

- *On-policy:* learn about policy currently executing
- How do we get rid of exploring starts?
  - The policy must be eternally *soft*:
    - $\pi(a|s) > 0$ for all s and a
  - e.g. ε-soft policy:
    - probability of an action = $\frac{\epsilon}{|\mathcal{A}(s)|}$ or $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$

non-max max (greedy)

- Similar to GPI: move policy *towards* greedy policy (e.g., ε-greedy)
- Converges to best ε-soft policy

On-policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list
$\pi(a|s) \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
(a) Generate an episode using π
(b) For each pair s, a appearing in the episode:
$G \leftarrow$ return following the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
(c) For each s in the episode:
$A^* \leftarrow \arg\max_a Q(s,a)$
For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

What we've learned about Monte Carlo so far

- MC has several advantages over DP:
  - Can learn directly from interaction with environment
  - No need for full models
  - No need to learn about ALL states (no bootstrapping)
  - Less harmed by violating Markov property
- MC methods provide an alternate policy evaluation process
- One issue to watch for: maintaining sufficient exploration
  - exploring starts, soft policies

Off-policy methods

- Learn the value of the *target policy* π from experience due to *behavior policy b*
- For example, π is the greedy policy (and ultimately the optimal policy) while *b* is exploratory (e.g., ε-soft)
- In general, we only require *coverage*, i.e., that *b* generates behavior that covers, or includes, π

$$\pi(a|s) > 0 \quad \text{for every } s, a \text{ at which } \; b(a|s) > 0$$

- Idea: *importance sampling*
  - Weight each return by the *ratio of the probabilities* of the trajectory under the two policies

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under π:

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}$$

- In importance sampling, each return is weighted by the relative probability of the trajectory under the two policies

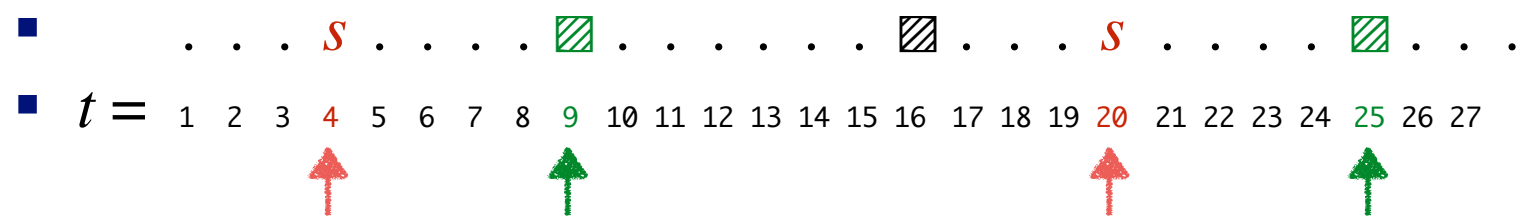
$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

- This is called the *importance sampling ratio*
- All importance sampling ratios have expected value 1

$$\mathbb{E}\left[\frac{\pi(A_k|S_k)}{b(A_k|S_k)}\right] \doteq \sum_a b(a|S_k)\frac{\pi(a|S_k)}{b(a|S_k)} = \sum_a \pi(a|S_k) = 1$$

Importance Sampling

- New notation: time steps increase across episode boundaries:



$\mathcal{T}(s) = \{4, 20\}$ set of start times

$T(4) = 9 \quad T(20) = 25$ next termination times

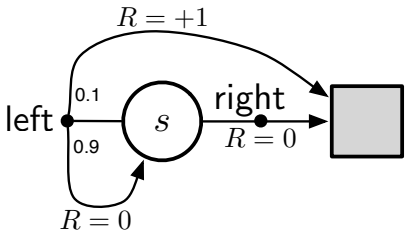
- *Ordinary importance sampling* forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

- Whereas *weighted importance sampling* forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Example of infinite variance under *ordinary* importance sampling



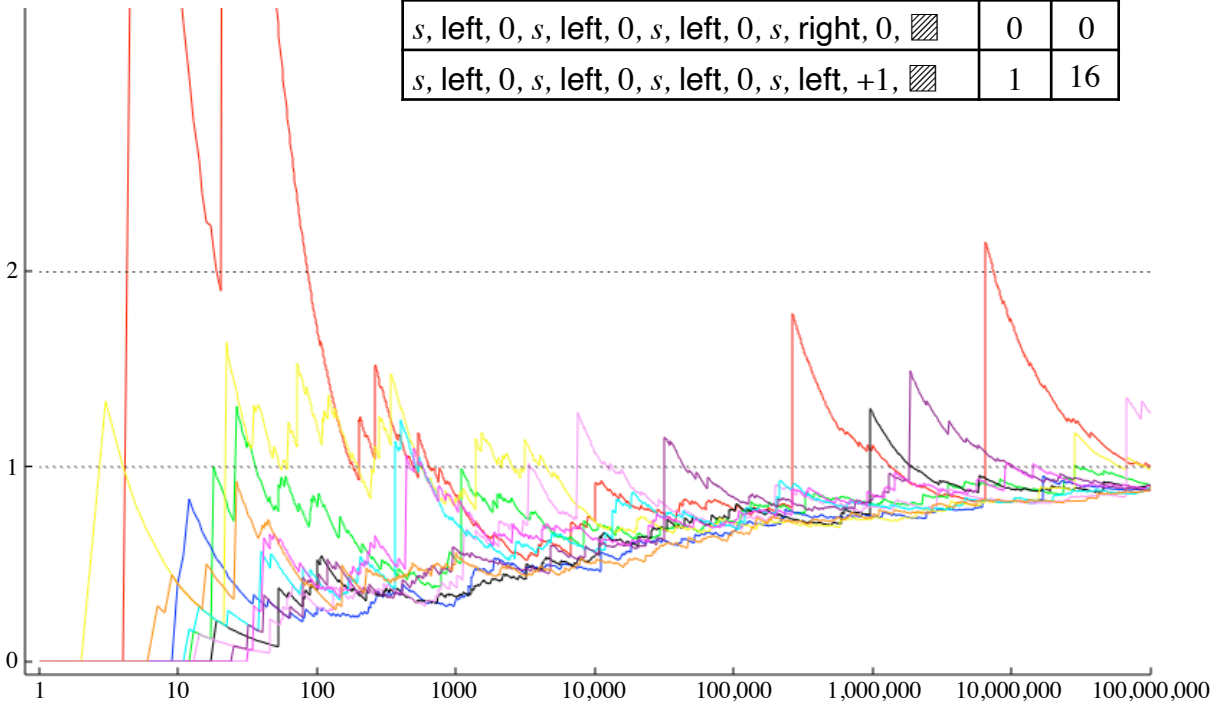
$\pi(\text{left}|s) = 1$ $\quad$ $\gamma = 1$

$b(\text{left}|s) = \frac{1}{2}$ $\quad$ $v_\pi(s) = 1$

$\frac{\pi(\text{right}|s)}{b(\text{right}|s)} =$ $\quad$ $\frac{\pi(\text{left}|s)}{b(\text{left}|s)} =$

Trajectory	G_0	$\rho_{0:T-1}$
s, left, 0, s, left, 0, s, left, 0, s, right, 0, ▨	0	0
s, left, 0, s, left, 0, s, left, 0, s, left, +1, ▨	1	16

Monte-Carlo estimate of $v_\pi(s)$ with ordinary importance sampling (ten runs)



OIS:

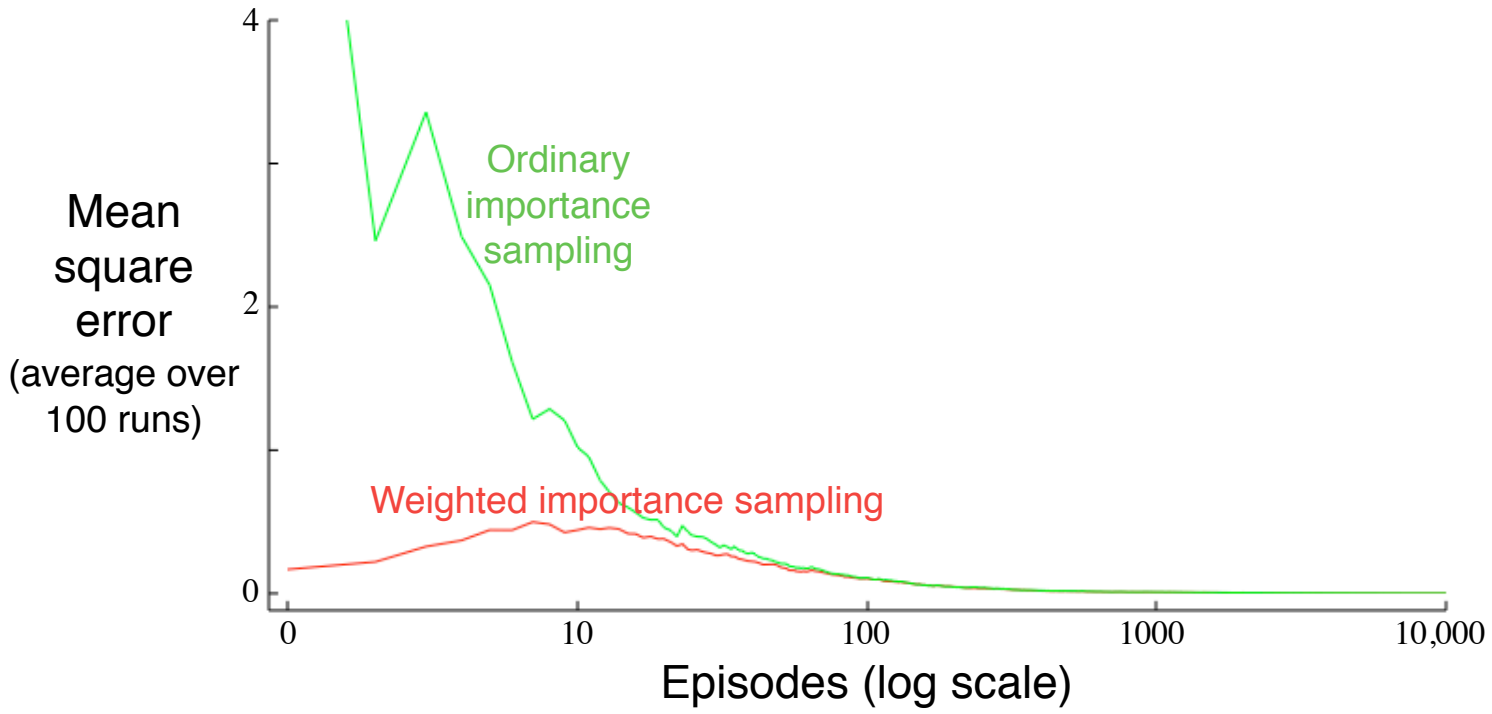
$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

WIS:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Example: Off-policy Estimation of the value of a *single* Blackjack State

- State is player-sum 13, dealer-showing 2, useable ace
- Target policy is stick only on 20 or 21
- Behavior policy is equiprobable
- True value ≈ –0.27726



Off-policy MC prediction, for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$

Repeat forever:
$b \leftarrow$ any policy with coverage of π
Generate an episode using b:
$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
$G \leftarrow 0$
$W \leftarrow 1$
For $t = T-1, T-2, \ldots$ downto 0:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
If $W = 0$ then ExitForLoop

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

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(S_t,a)$ (with ties broken consistently)

Repeat forever:
$b \leftarrow$ any soft policy
Generate an episode using b:
$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
$G \leftarrow 0$
$W \leftarrow 1$
For $t = T-1, T-2, \ldots$ downto 0:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then ExitForLoop
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Target policy is greedy and deterministic

Behavior policy is soft, typically ε-greedy

Discounting-aware Importance Sampling (motivation)

- So far we have weighted returns without taking into account that they are a discounted sum
- This can't be the best one can do!
- For example, suppose $\gamma = 0$
  - Then G_0 will be weighted by

$$\rho_{0:T-1} = \frac{\pi(A_0|S_0)}{b(A_0|S_0)} \frac{\pi(A_1|S_1)}{b(A_1|S_1)} \cdots \frac{\pi(A_{T-1}|S_{T-1})}{b(A_{T-1}|S_{T-1})}$$

  - But it really need only be weighted by

$$\rho_{0:1} = \frac{\pi(A_0|S_0)}{b(A_0|S_0)}$$

  - Which would have much smaller variance

Prediction vs Control

- For the control problem (finding an optimal policy), Dynamic Programming (DP) and Monte Carlo (MC) methods both use some variation of **generalized policy iteration (GPI)**.
- The differences in the methods are primarily differences in their approaches to the prediction problem.

Bootstrapping and Sampling

Bootstrapping: update involves an estimate of the value function

- DP methods bootstrap
- MC methods **do not** bootstrap

Sampling: update **does not** involve an expected value

- MC method sample
- Classical DP **does not** sample

Prediction vs Control

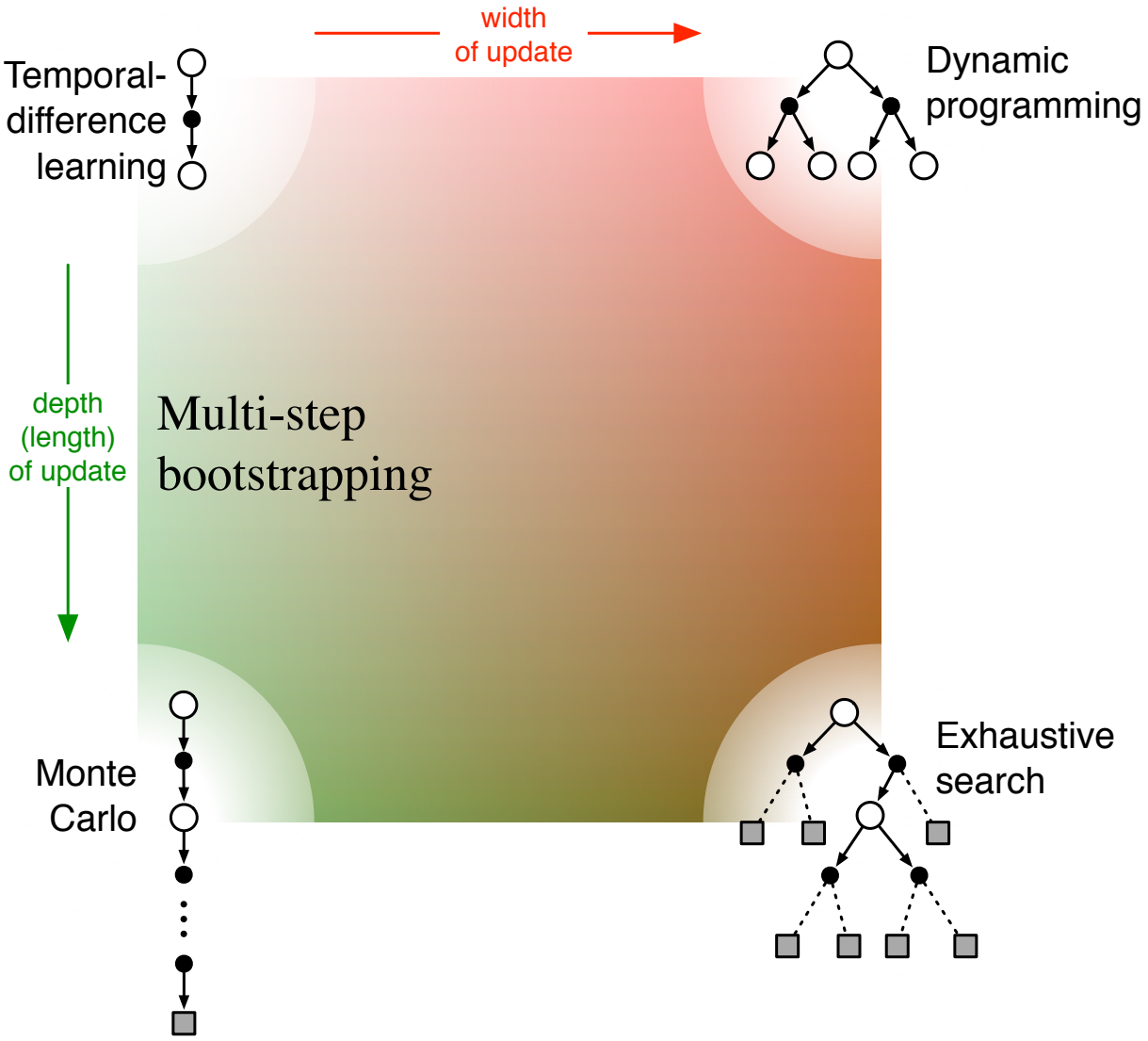
Dynamic Programming:

- Explore the given dynamics model to learn a value function
- Iterate until V or Q or π converge using explicit P(s'|s,a)

Monte Carlo and Temporal Differences:

- Learn from experience following a policy
- Update estimate V of v_π for the nonterminal states S_t that were seen in the sampled episode
- Sample from P(s'|s,a) until goal state, then update backwards

Unified View



Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment
  - No need for full models
  - Less harmed by violating Markov property
- MC methods provide an alternate policy evaluation process
- One issue to watch for: maintaining sufficient exploration
  - exploring starts, soft policies
- Introduced distinction between *on-policy* and *off-policy* methods
- Introduced *importance sampling* for off-policy learning
- Introduced distinction between *ordinary* and *weighted* IS
- Introduced two *return-specific* ideas for reducing IS variance
  - *discounting-aware* and *per-reward* IS