Winter School 2024
Reinforcement Learning

Prediction

Dr. Lorenzo Brigato

Artificial Intelligence in Health and Nutrition (AIHN) Laboratory
ARTORG Center for Biomedical Engineering Research
University of Bern

Outlook

- Model-based prediction (via Dynamic Programming)
- Monte Carlo Learning
- Temporal Difference (TD) Learning
- N-Step TD

Policy Evaluation

- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_1, \dots, S_k \sim \pi$$

- Known MDP (model available)
  - Dynamic Programming
- Unknown MDP (model not available)
  - Monte-Carlo Learning
  - Temporal-Difference Learning
  - N-step TD

Planning by Dynamic Programming

- Dynamic programming assumes full knowledge of the MDP, and it is used for planning:
- For Prediction:
  - Input: MDP/MRP and policy π
  - Output: State-value function
- For Control:
  - Input: MDP
  - Output: optimal value function and optimal policy

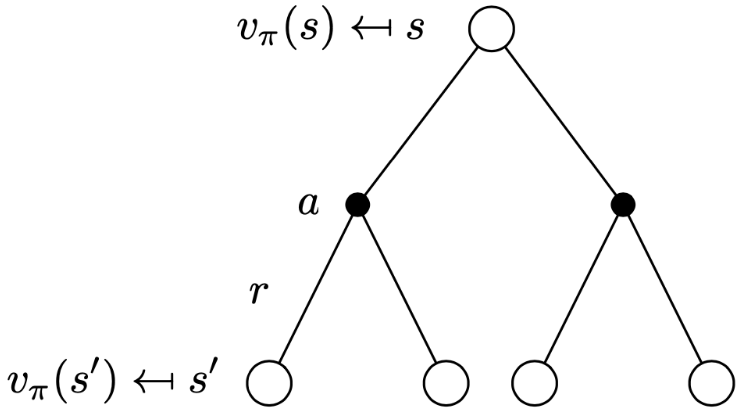
Iterative Policy Evaluation

- Problem: evaluate a given policy π
- Solution: iterative application of Bellman expectation update

$$\mathrm{v}_1 \rightarrow \mathrm{v}_2 \rightarrow \cdots \rightarrow \mathrm{v}_\pi$$

- Using synchronous updates,
  - At each iteration $k + 1$
  - For all states $s \in \mathcal{S}'$
  - Update $\mathrm{v}_{k+1}(s)$ from $\mathrm{v}_k(s')$
  - where s′ is a successor state of s
- Convergence to v_π is proven at a geometric rate (refer to Banach's fixed-point theorem)

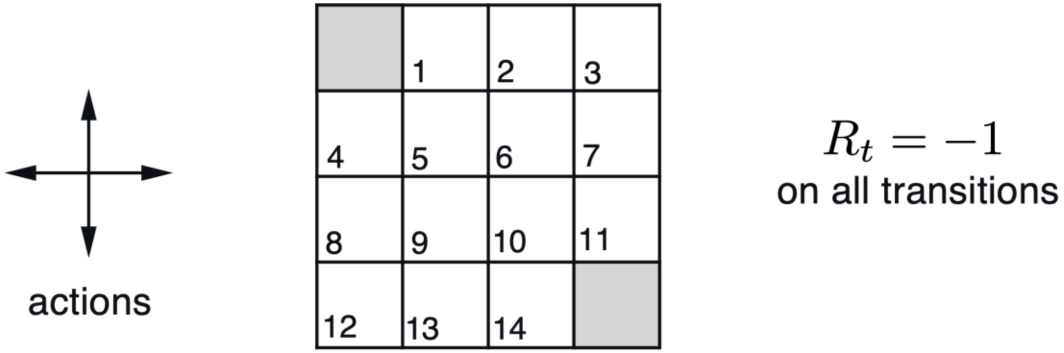
Iterative Policy Evaluation



$$\mathrm{v}_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \mathrm{v}_k(s') \right)$$

[An Introduction to Reinforcement Learning, Sutton and Barto]

Evaluating a Random Policy in the Small Gridworld



- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states 1, ..., 14
- Terminal states shown as shaded squares
- Actions leading out of the grid leave state unchanged
- Reward is −1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(u \mid \cdot) = \pi(d \mid \cdot) = \pi(r \mid \cdot) = \pi(l \mid \cdot) = 0.25$$

[An Introduction to Reinforcement Learning, Sutton and Barto]

Iterative Policy Evaluation in Small Gridworld (1/2)

v_k for the random policy

$k = 0$

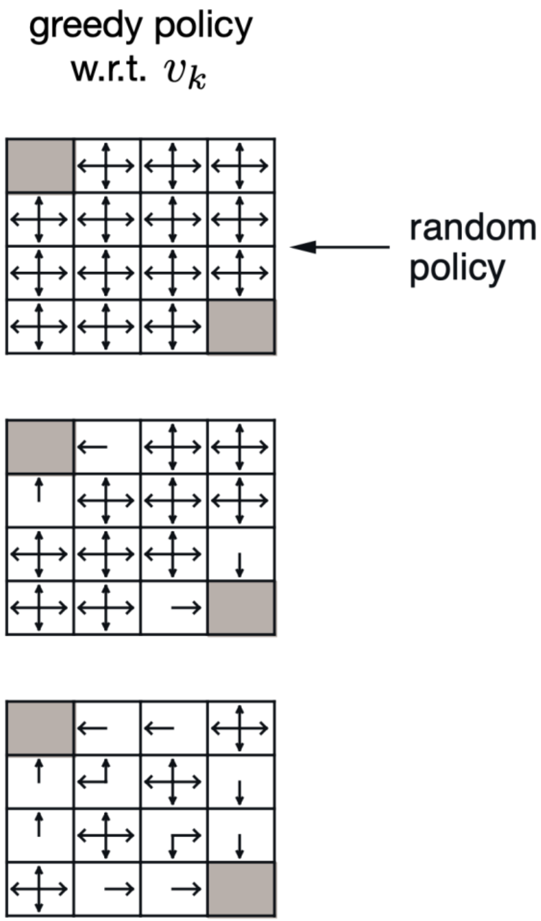
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

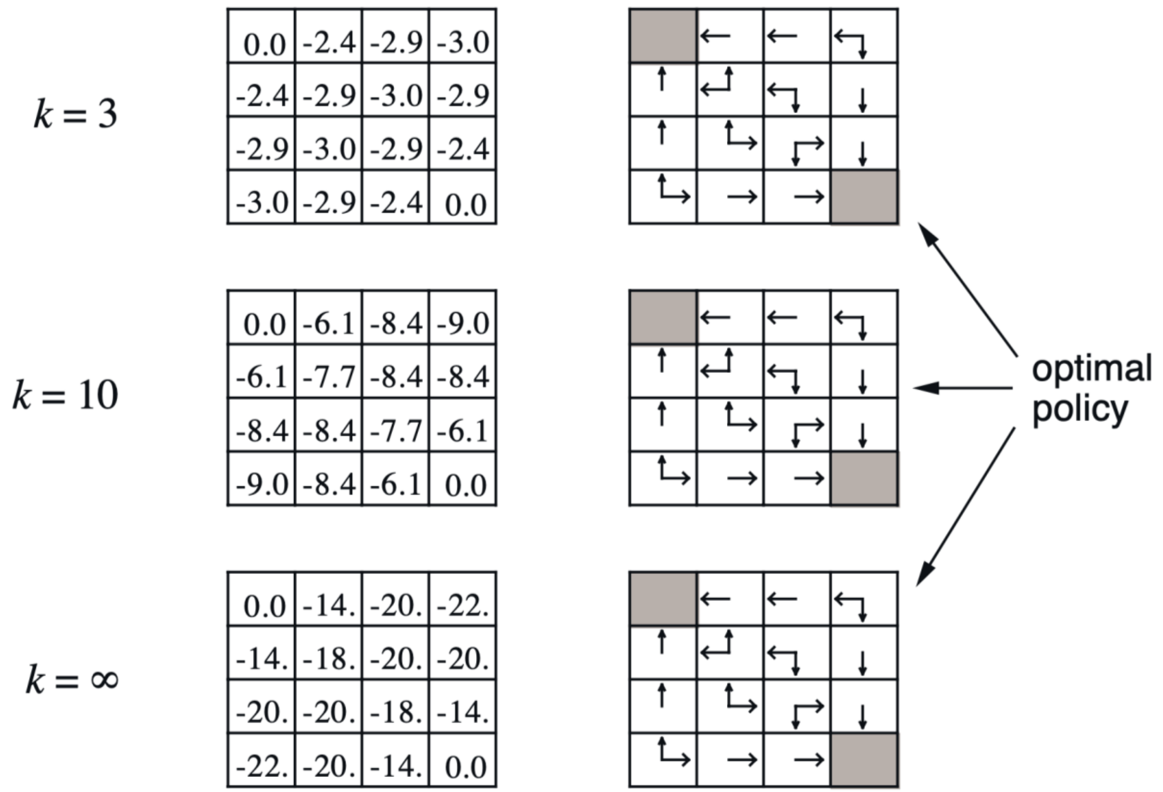
$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



[An Introduction to Reinforcement Learning, Sutton and Barto]

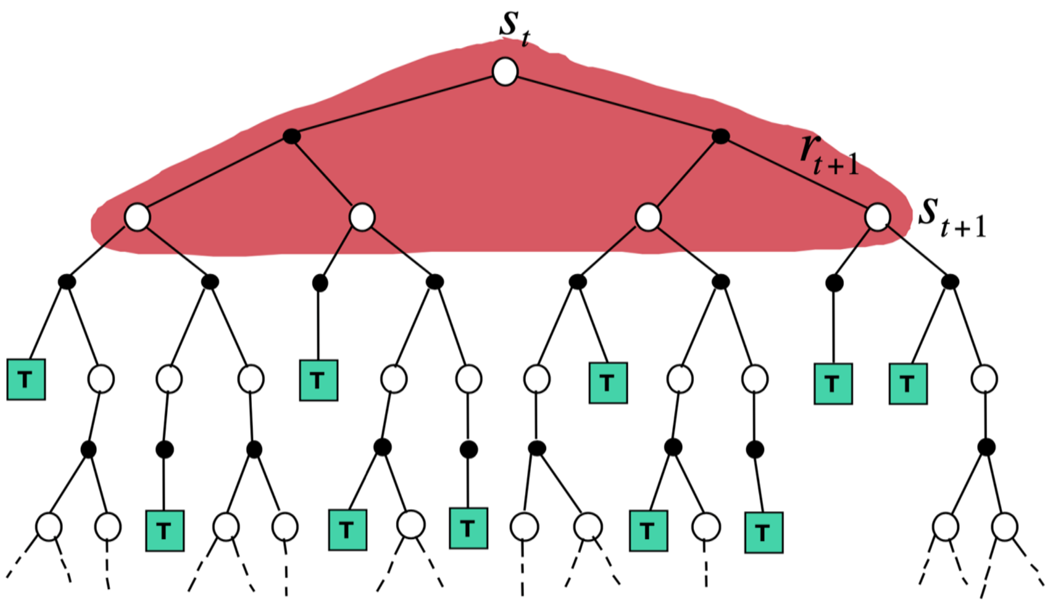
Iterative Policy Evaluation in Small Gridworld (2/2)



[An Introduction to Reinforcement Learning, Sutton and Barto]

Dynamic Programming Update

$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$



[David Silver, IRL, UCL 2015]

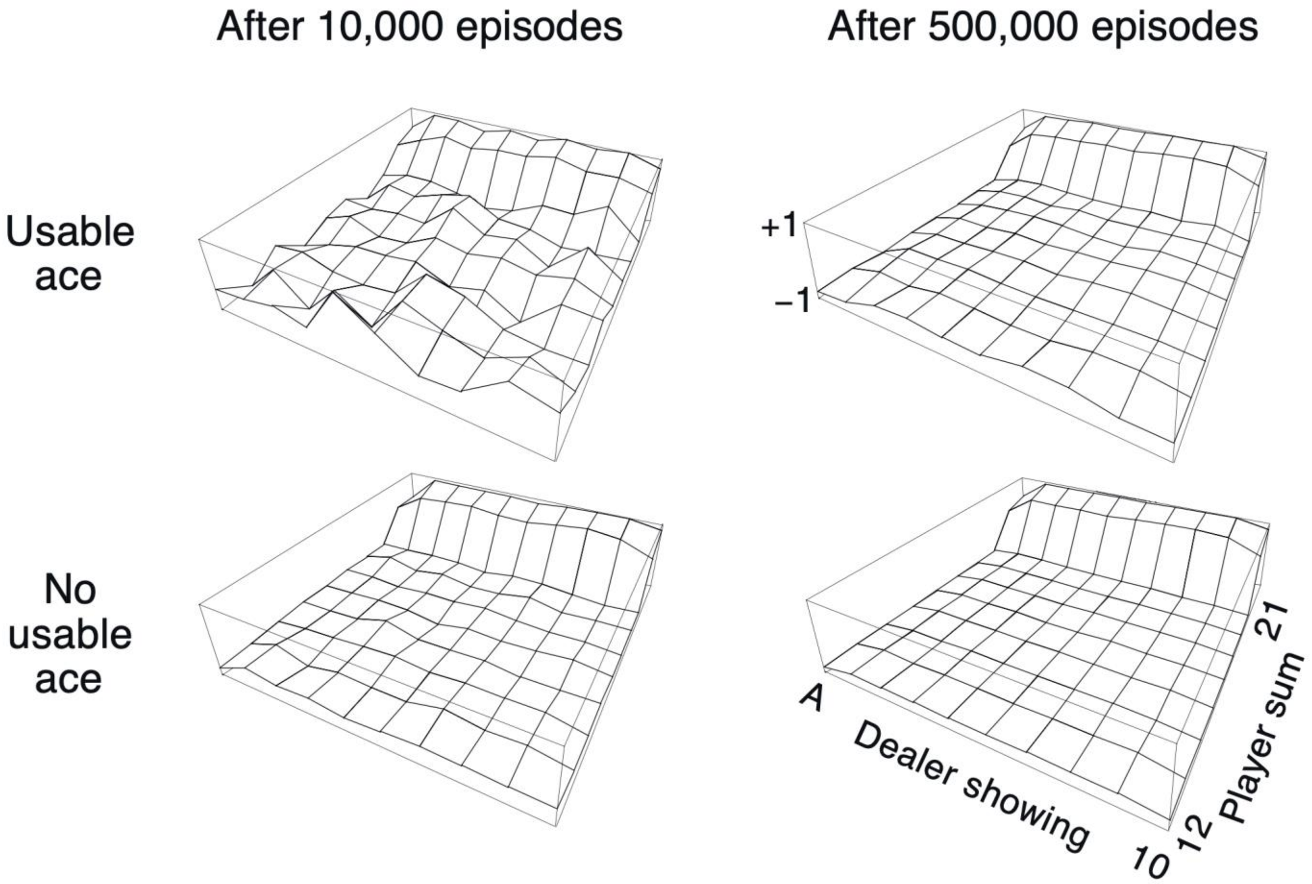
Model-Free Policy Evaluation

- Monte-Carlo (MC) Learning
  - MC methods learn from complete episodes of experience
    - Simplest possible idea: value = mean return
  - Caveat: can only apply MC to episodic MDPs
    - All episodes must terminate
- Temporal-Difference (TD) Learning
  - TD methods learn from incomplete episodes of experience
    - TD learns from episodes, by bootstrapping (updates a guess towards a guess)

Example: Blackjack

- States (200 in total):
  - Current sum (12-21)
  - Dealer's showing card (ace or 2-10)
  - Do I have a "useable" ace? (yes-no)
- Actions
  - *hit*: Take another card (no replacement)
  - *stick*: Stop receiving cards (and terminate)
- Rewards
  - for stick:
    - +1 if sum of cards > sum of dealer cards
    - 0 if sum of cards = sum of dealer cards
    - -1 if sum of cards < sum of dealer cards
  - for hit:
    - -1 if sum of cards > 21 (and terminate)
    - 0 otherwise
- Policy: stick if sum of cards ≥ 20, otherwise hit

Blackjack Value Function after MC Learning



[An Introduction to Reinforcement Learning, Sutton and Barto]

Incremental Mean

– The mean μ_k of a sequence $x_1, x_2, \ldots, x_k$ can be computed incrementally,

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\
&= \frac{1}{k}\left(x_k + \sum_{i=1}^{k-1} x_i\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}$$

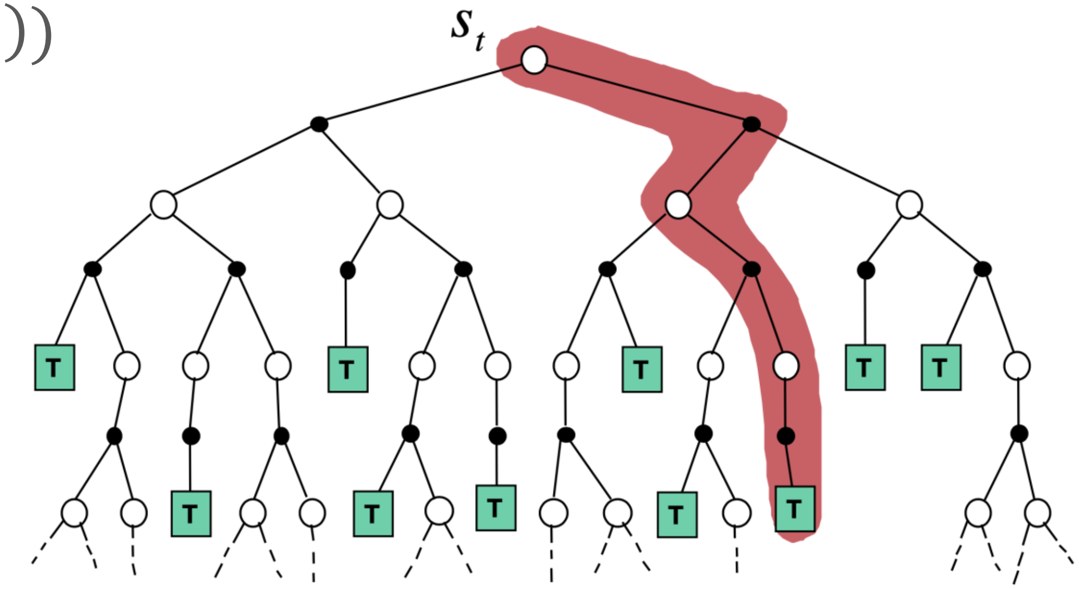
MC Incremental Update

- Update $V(s)$ incrementally after episode $S_1, A_1, R_1, \ldots, S_k$
- For each state S_t with return G_t and number of visits $N(S_t)$:
  - Update state value

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

  - Update visit count

$$N(s) \leftarrow N(s) + 1$$



[David Silver, IRL, UCL 2015]

Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is model-free: no knowledge of MDP transitions / rewards
- TD learns from incomplete episodes, by bootstrapping
- TD updates a guess towards a guess

MC and TD

- Goal: learn v_π from episodes of experience under policy π
- Incremental Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t

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

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$

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

  - $R_{t+1} + \gamma V(S_{t+1})$ is called TD target
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error

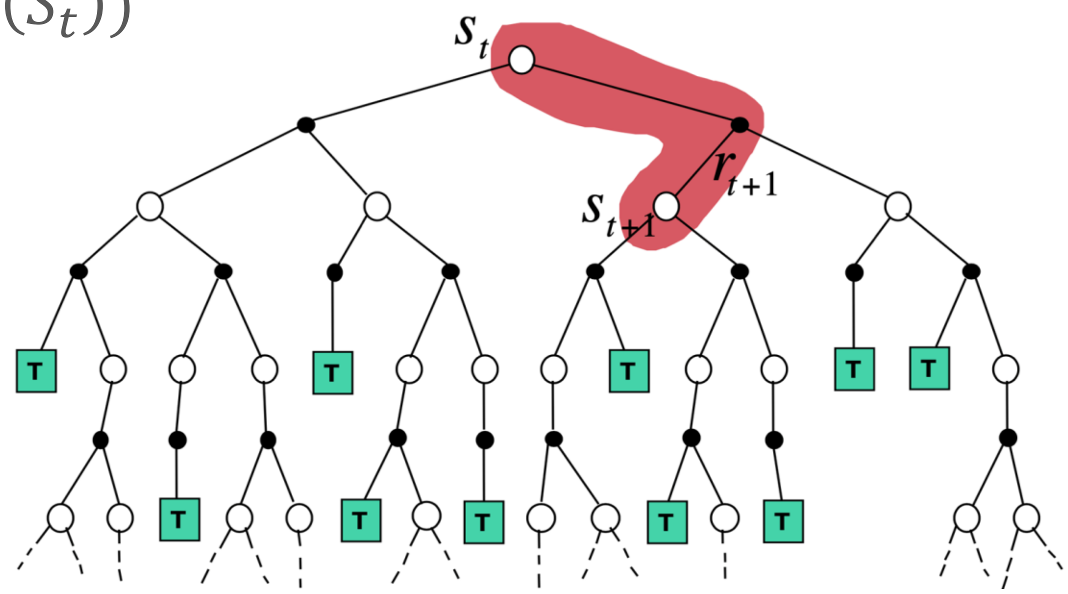
TD Incremental Update

- Update $V(s)$ incrementally after transition $S_t, A_t, R_{t+1}, S_{t+1}$
- For each state S_t with reward R_{t+1}:
  - Update state value toward estimated return $R_{t+1} + \gamma V(S_{t+1})$

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

$R_{t+1} + \gamma V(S_{t+1})$ is called TD target

$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error



[David Silver, IRL, UCL 2015]

Advantages and Disadvantages of MC vs TD (1/3)

- TD can learn before knowing the outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn without the outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

Advantages and Disadvantages of MC vs TD (2/3)

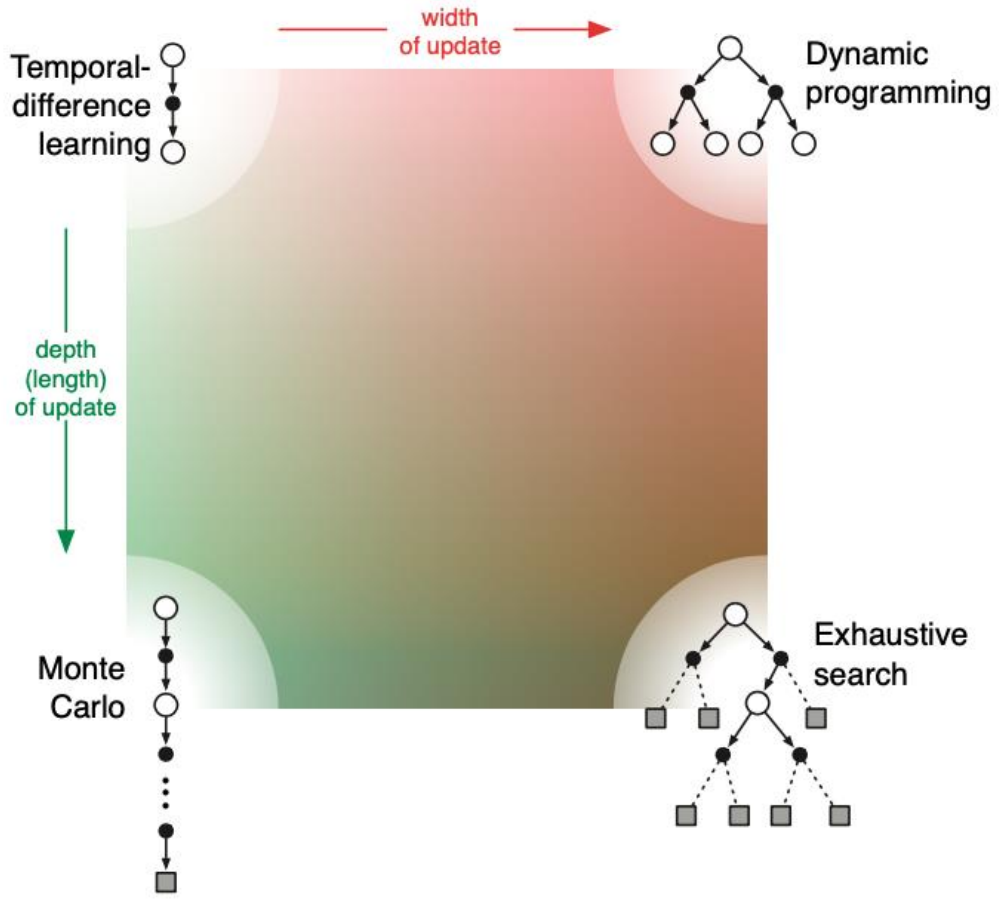
- MC has high variance, zero bias
  - Good convergence properties
  - Not very sensitive to initial value
  - Very simple to understand and use
- TD has low variance, some bias
  - Usually more efficient than MC
  - TD(0) converges to $v_{\pi}(s)$
  - More sensitive to initial value

Advantages and Disadvantages of MC vs. TD (3/3)

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

Unified View of Reinforcement Learning

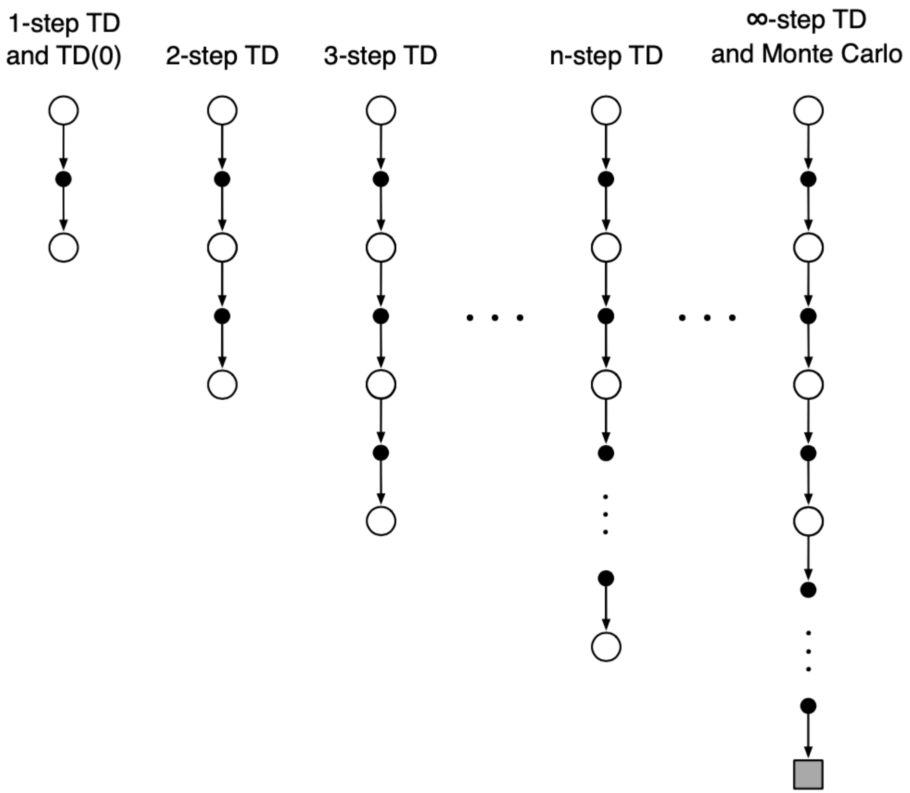
- **Bootstrapping**: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- **Sampling**: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples



[An Introduction to Reinforcement Learning, Sutton and Barto]

n-Step Prediction

– Instead of just looking one step in the future, let's look n steps:



[An Introduction to Reinforcement Learning, Sutton and Barto]

n-Step Return

- Consider the following n-step returns for n=1,2,…,∞
  - $n = 1 - TD$

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

  - $n = 2$

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+1} + \gamma^2 V(S_{t+2})$$

  - $n = \infty$ - MC

$$G_{t:t+\infty} = R_{t+1} + \gamma R_{t+1} + \ \ldots$$

- We can define the n-step return

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+1} + \cdots + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha(G_{t:t+n} - V(S_t))$$

Example: Random Walk (19-states)