Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Reinforcement Learning

—

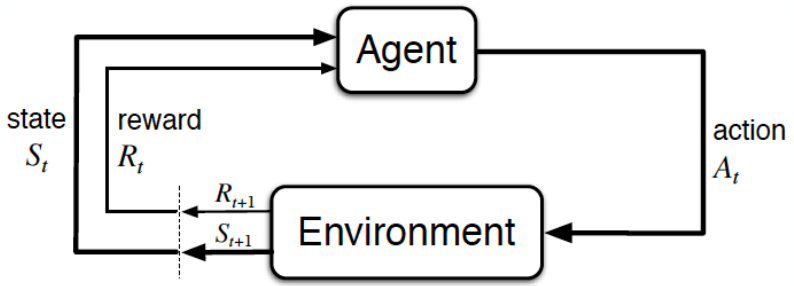
Lecture 3: Model-free Prediction

Christopher Mutschler

Recap

Markov Decision Processes

- Agent learns by interacting with an environment over many time-steps:
- Markov Decision Process (MDP) is a tool to formulate RL problems
  - Description of an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Note:
If the interaction does stop at some point in time (T) then we have an *episodic RL problem.*

- At each step t, the agent:
  - is at state $\mathrm{S_t}$,
  - performs action $\mathrm{A_t}$,
  - receives reward $\mathrm{R_t}$.

- At each step t, the environment:
  - receives action $\mathrm{A_t}$ from the agent,
  - provides reward $\mathrm{R_t}$,
  - moves at state $\mathrm{S_{t+1}}$,
  - increments time $\mathrm{t} \leftarrow \mathrm{t} + 1$.

Recap

Markov Decision Processes

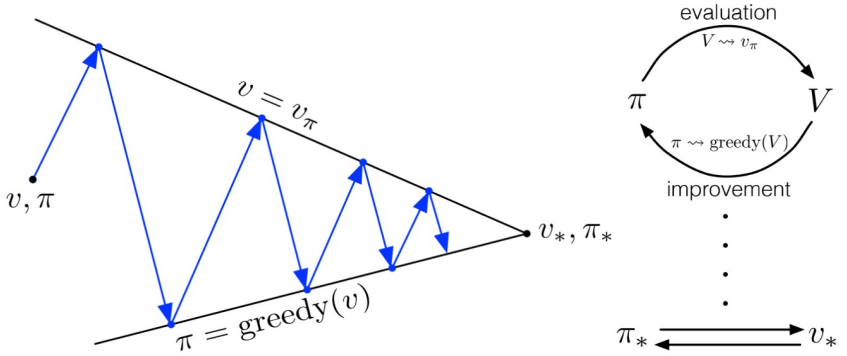
- The general RL setting
  - Agent-Environment-Interface: actions, states, and rewards
  - Agent interacts with the environment over a sequence of discrete time steps (episodic or continual)
  - Policy as a stochastic rule to select actions

- MDPs as tools to describe RL problems
  - Main ingredients: states, actions, state transition probabilities, return, and discount
  - Value functions that describe the expected return following a particular policy
  - Bellman equation as expression of the relationship between the value of a state and the value of its successor states

Recap
Dynamic Programming

- Dynamic Programming (DP) methods to find optimal controllers
  - DP methods are guaranteed to find optimal solutions for Q and V in polynomial time (in number of states and actions) and are exponentially faster than direct search
  - Policy Iteration computes the value function under a given policy to improve the policy while value iteration directly works on the states
    - Perform sweeps through the state set
    - Implement the Bellman equation update
    - Use bootstrapping
  - Require complete and accurate model of the environment
  - Have limited applicability in practice…

    → as they need to know the dynamics of the environment!



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- So far: We know our MDP model $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$.
  - Planning by using dynamic programming
  - Solve a known MDP

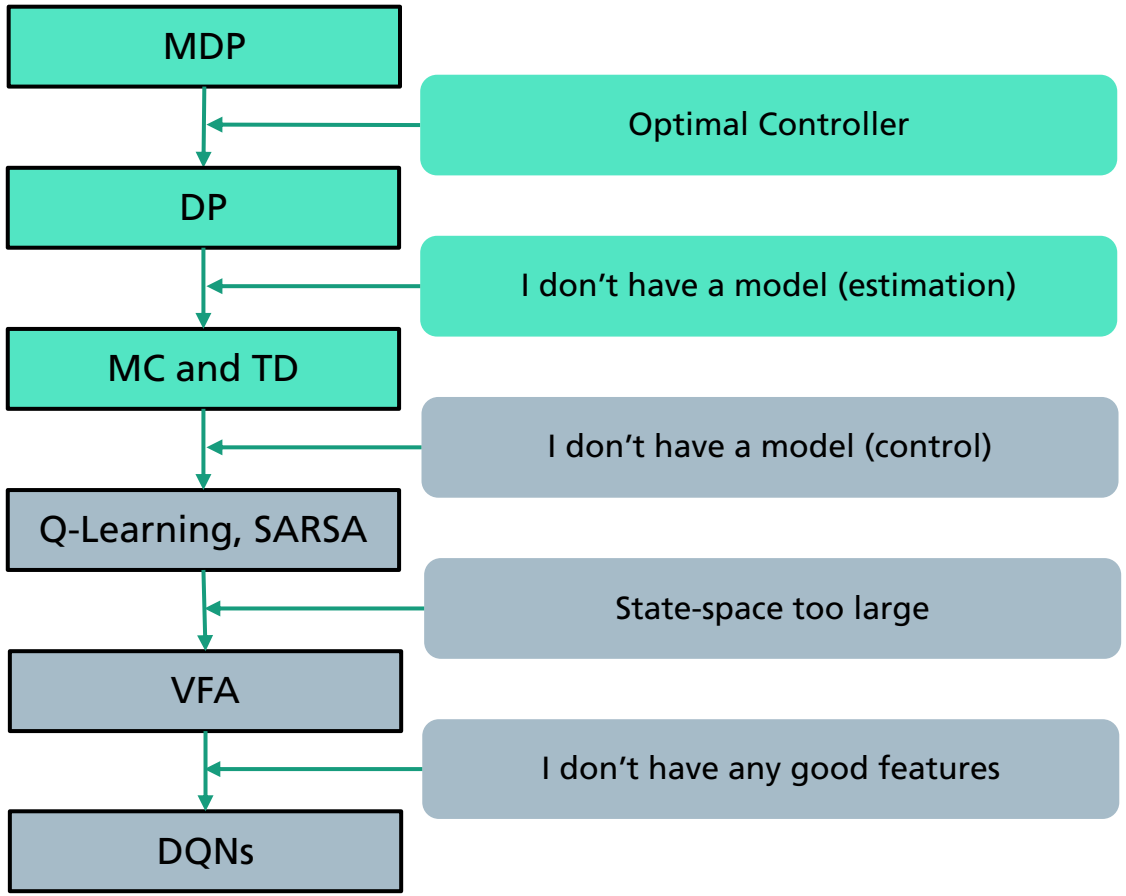
- What if we don't know the model, i.e., $\mathcal{P}$ or $\mathcal{R}$ or both?

- We distinguish between 2 problems for unknown MDPs:

  - **Model-free Prediction:** Evaluate the future, given the policy π.
    (estimate the value function)

  - **Model-free Control:** Optimize the future by finding the best policy π.
    (optimize the value function)

Overview


MDP
Optimal Controller
DP
I don’t have a model (estimation)
MC and TD
I don’t have a model (control)
Q-Learning, SARSA
State-space too large
VFA
I don’t have any good features
DQNs

Monte Carlo and TD Methods

Assumptions

- We know that the model of the world can be described by an MDP:

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$$

- We know the (discrete) state and action spaces, i.e., $\mathcal{S}$ and $\mathcal{A}$.
- We can interact with the world (with some policy π).
- We receive experience samples from the environment in the form

$$(S_t, A_t, R_t, S_{t+1}) = (s, a, r, s').$$

Monte Carlo and TD Methods

- Idea:
  - Use the samples to estimate the true V- and Q-value functions for the policy π:

$$V^{\pi}(s)$$
$$Q^{\pi}(s,a)$$

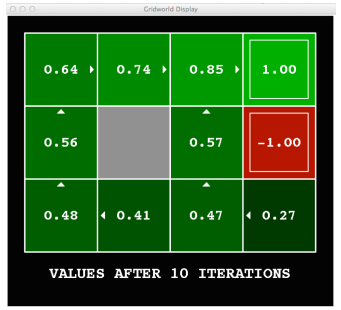
  - Use value function estimations for model-free prediction:

$$V(s) \approx V^{\pi}(s)$$
$$Q(s,a) \approx Q^{\pi}(s,a).$$

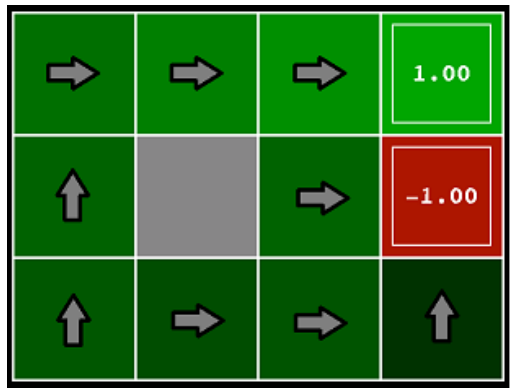
  - Two policy evaluation approaches:
    - Monte Carlo (MC) Learning
    - Temporal Difference (TD) Learning
    - variants in between, i.e., TD(λ)

Monte Carlo and TD Methods

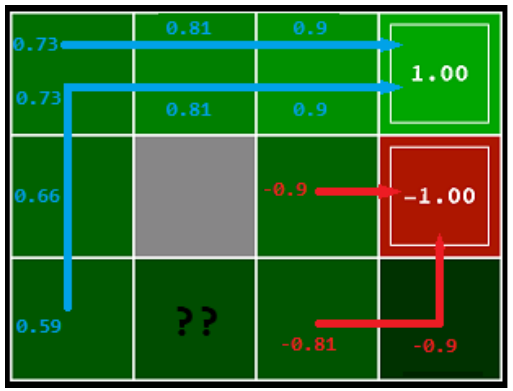
Remember:



- Idea:
  - Use the samples to estimate the true V- and Q-value functions for the policy π



Given Policy



Randomly select state and follow policy
&
Compute discounted return for each state

Average the values on each state

https://medium.com/@zsalloum/monte-carlo-in-reinforcement-learning-the-easy-way-564c53010511

Monte Carlo Policy Evaluation

- MC Policy Evaluation
  - MC methods learn from episodes of experience under policy π:

$$s_t, a_t, r_t, s_{t+1}, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T \sim \pi$$

  - To evaluate a state $s \in \mathcal{S}$ we keep track of the rewards received from that state onwards.

- First-Visit Monte-Carlo Policy Evaluation:
  - First time-step t that state s is visited in an episode
    - Increment counter $N(s) \leftarrow N(s) + 1$,
    - Increment total return $S(s) \leftarrow S(s) + G_t$,
    - Value is estimated by mean return: $V(s) = S(s)/N(s)$
  - Our estimation $V(s)$ will come close to $V^\pi(s)$ as $N(s) \rightarrow \infty$.
    (considering the law of large numbers)

Monte Carlo Policy Evaluation

- MC Policy Evaluation
  - MC methods learn from episodes of experience under policy π:

$$s_t, a_t, r_t, s_{t+1}, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T \sim \pi$$

  - To evaluate a state $s \in \mathcal{S}$ we keep track of the rewards received from that state onwards.

- Every-Visit Monte-Carlo Policy Evaluation:
  - Every time-step t that state s is visited in an episode
    - Increment counter $N(s) \leftarrow N(s) + 1$,
    - Increment total return $S(s) \leftarrow S(s) + G_t$,
    - Value is estimated by mean return: $V(s) = S(s)/N(s)$
  - Our estimation $V(s)$ will come close to $V^\pi(s)$ as $N(s) \rightarrow \infty$.
    (considering the law of large numbers)

Monte Carlo Policy Evaluation

- MC Policy Evaluation

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

Input: a policy π 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 π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

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

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo Policy Evaluation

Exercise 5.5 Consider an MDP with a single nonterminal state and a single action that transitions back to the nonterminal state with probability p and transitions to the terminal state with probability $1-p$. Let the reward be $+1$ on all transitions, and let $\gamma = 1$. Suppose you observe one episode that lasts 10 steps, with a return of 10. What are the first-visit and every-visit estimators of the value of the nonterminal state? □

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

Input: a policy π 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 π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average$(Returns(S_t))$

First-Visit MC: $v_s = 10$

Every-Visit MC: $v_s = \frac{1}{10} \cdot (10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1) = 5.5$

Monte Carlo Policy Evaluation

Example: Blackjack. MDP:

- States:
  - Current sum (12-21) [$\mathcal{P}$ models an automatic twist if sum of cards < 12]
  - Dealer's showing card (ace-10)
  - Do I have a usable ace (yes or no)
- Actions:
  - Stick: stop receiving cards (and terminate)
  - Twist: take another card (no replacement)
- Rewards:
  - 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
  - Twist:
    - -1 if sum of cards > 21 (and terminate), 0 otherwise

source: shutterstock.com

Monte Carlo Policy Evaluation

Example: Blackjack

- π: stick if sum of cards $\geq$ 20 (i.e., 20 or 21), otherwise twist.
- No discounting.

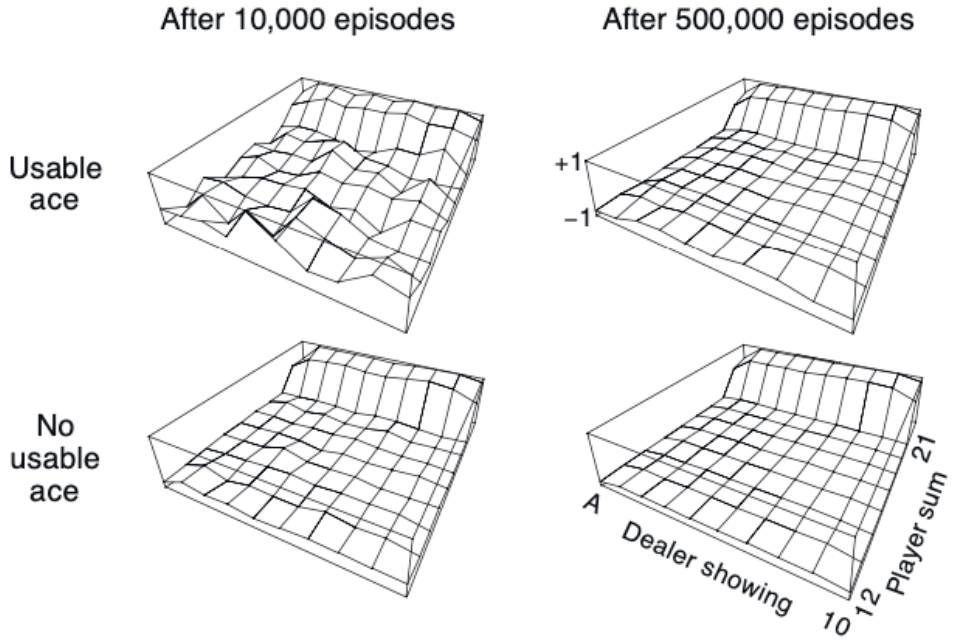


Figure 5.1: Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation. ■

Monte Carlo Policy Evaluation

Example: Blackjack

- π: stick if sum of cards $\geq$ 20 (i.e., 20 or 21), otherwise twist.
- No discounting.

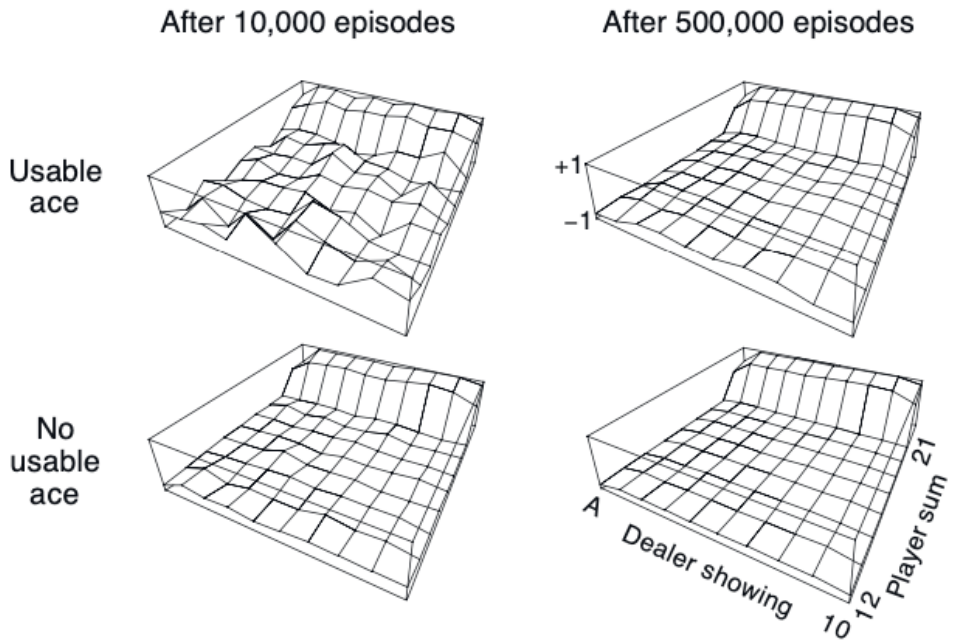


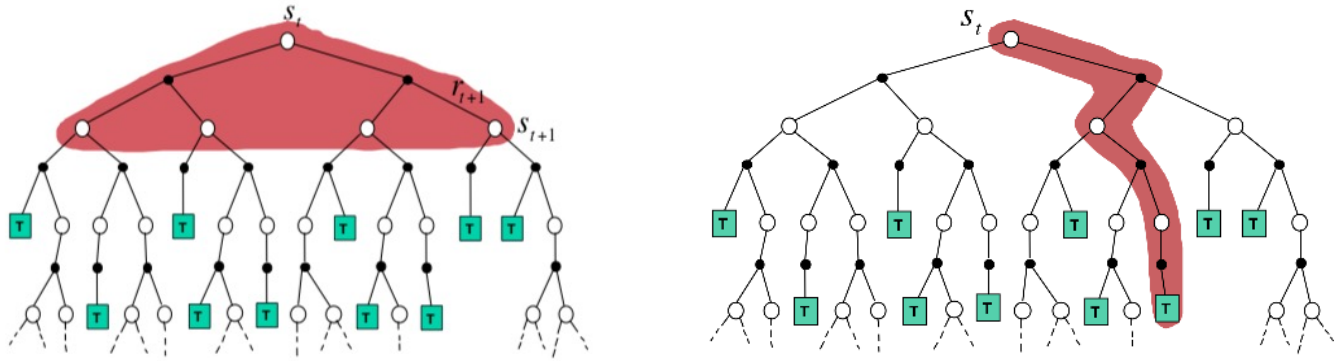
Figure 5.1: Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation. ■

Exercise 5.1 Consider the diagrams on the right in Figure 5.1. Why does the estimated value function jump up for the last two rows in the rear? Why does it drop off for the whole last row on the left? Why are the frontmost values higher in the upper diagrams than in the lower? □

Exercise 5.2 Suppose every-visit MC was used instead of first-visit MC on the blackjack task. Would you expect the results to be very different? Why or why not? □

Monte Carlo Policy Evaluation

- Backup Diagrams compared to DP:



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo Policy Evaluation

- MC Policy Evaluation
- **Incremental Mean**: the mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally:

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\,\mu_{k-1}\right) \\
&= \frac{1}{k}\left(x_k + k u_{k-1} - u_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

Monte Carlo Policy Evaluation

- MC Policy Evaluation
- **Incremental Monte-Carlo Updates**

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j$$
$$= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right)$$
$$= \frac{1}{k}\left(x_k + (k-1)\,\mu_{k-1}\right)$$
$$= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)$$

- Update $V(s)$ incrementally after each episode.
- For each state s with actual return G:

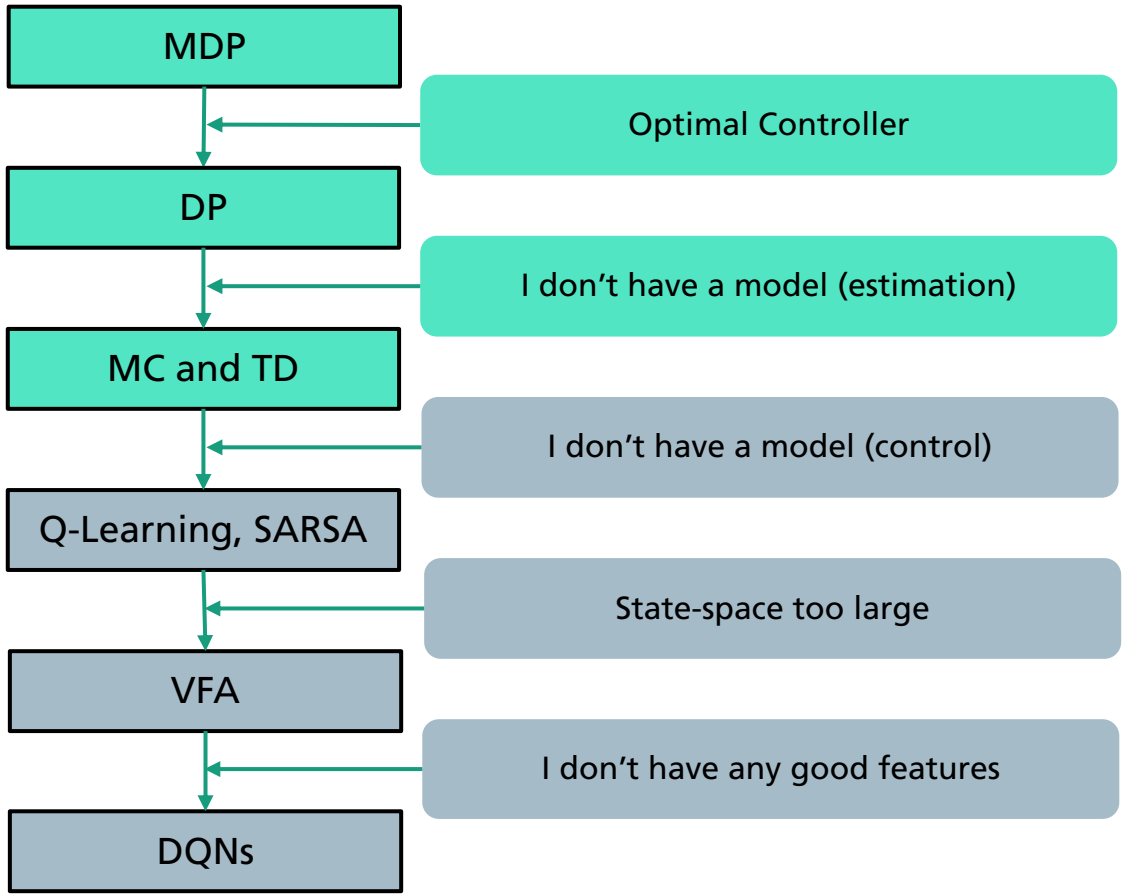
$$N(s) \leftarrow N(s) + 1 \quad \text{(just increment visit counter)}$$
$$V(s) \leftarrow V(s) + \frac{1}{N(s)}\left(G - V(s)\right) \quad \text{(update a bit → reduce error)}$$

- In non-stationary problems, it can be useful to track a running mean, i.e., forget old episodes:

$$V(s) \leftarrow V(s) + \alpha\left(G - V(s)\right).$$

$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$

Overview


MDP
Optimal Controller
DP
I don’t have a model (estimation)
MC and TD
I don’t have a model (control)
Q-Learning, SARSA
State-space too large
VFA
I don’t have any good features
DQNs

Monte Carlo and TD Methods

Assumptions:

- We know that the model of the world can be described by an MDP:

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$$

- We know the (discrete) state and action spaces, i.e., $\mathcal{S}$ and $\mathcal{A}$.
- We can interact with the world (with some policy π).
- We receive experience samples from the environment in the form

$$(S_t, A_t, R_t, S_{t+1}) = (s, a, r, s').$$

Monte Carlo and TD Methods

- Temporal-Difference Learning
  - Breaks up episodes and makes use of the intermediate returns
  - Learns directly from experience and interaction with the environment
  - Model-free: no knowledge of MDP
  - Learns from incomplete episodes (bootstrapping)
  - **We update a guess towards a guess**

Monte Carlo and TD Methods

- Temporal-Difference Learning: Idea of TD(0) Policy Evaluation

$$V^{\pi}(s) = \underbrace{r(s,\pi(s))} + \underbrace{\gamma \sum_{s' \in S} \mathcal{P}(s'|s,\pi(s)) V^{\pi}(s')}$$

We don't know the transition model

$$(s, a, r, s')$$

But we have real transitions available

$$V^{\pi}(s) = r + \gamma V^{\pi}(s')$$

Let's assume that the reality is the transition we observed

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$$

→ and update our old estimate "a bit" in this direction

Monte Carlo and TD Methods

- TD(0) vs. MC Policy Evaluation
  - Goal: learn value function v_π online from experience when we follow policy π

- Simplest TD learning algorithm: TD(0)
- Update value **towards estimation $\hat{G}$**:

$$V(s) \leftarrow V(s) + \alpha(\hat{\boldsymbol{G}} - V(s))$$
$$\hat{\boldsymbol{G}} = \boldsymbol{r} + \boldsymbol{\gamma V(s')}$$ (estimated return)

- $\hat{G}$ is called the TD target
- $\hat{G} - V(s)$ is called the TD error.

- Update $V(s)$ incrementally after each episode.
- For each state s with **actual return G**:

$$N(s) \leftarrow N(s) + 1$$ (just increment visit counter)
$$V(s) \leftarrow V(s) + \frac{1}{N(s)}\left(\boldsymbol{G} - V(s)\right)$$ (update a bit → reduce error)

- In non-stationary problems, it can be useful to track a running mean, i.e., forget old episodes:

$$V(s) \leftarrow V(s) + \alpha\left(\boldsymbol{G} - V(s)\right).$$

$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$

Monte Carlo and TD Methods

- TD(0) vs. MC Policy Evaluation

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

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

Input: a policy π 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 π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average$(Returns(S_t))$

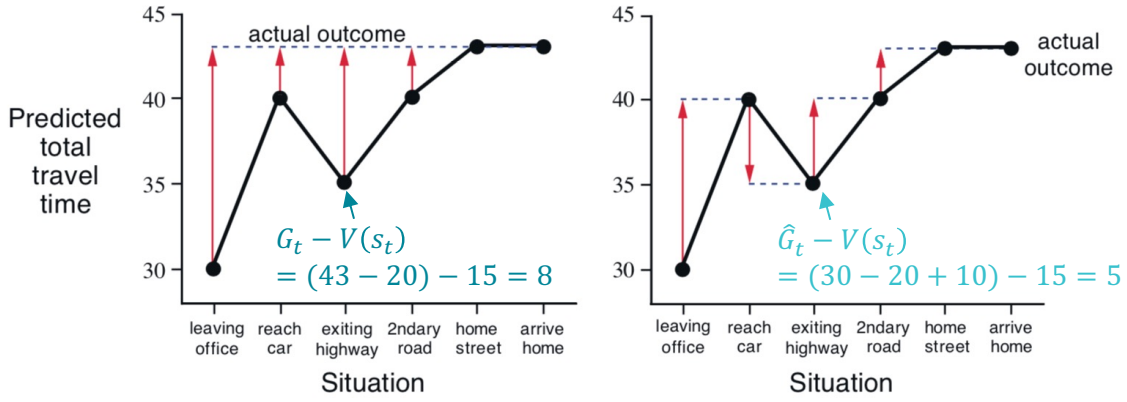
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Which one should I use? Does it make any difference?
- Example: Driving Home from work

State	*Elapsed Time [min]*	*Predicted Time to Go [min]*	*Predicted Total Time [min]*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

MC ($\alpha = 1$)

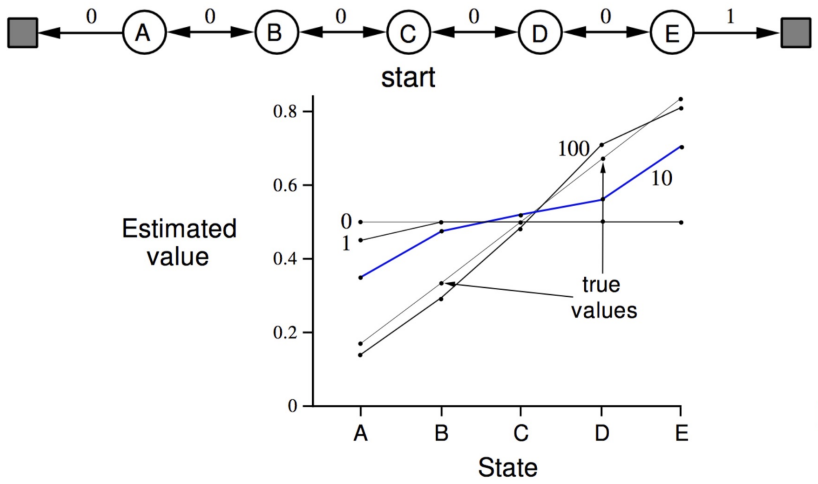


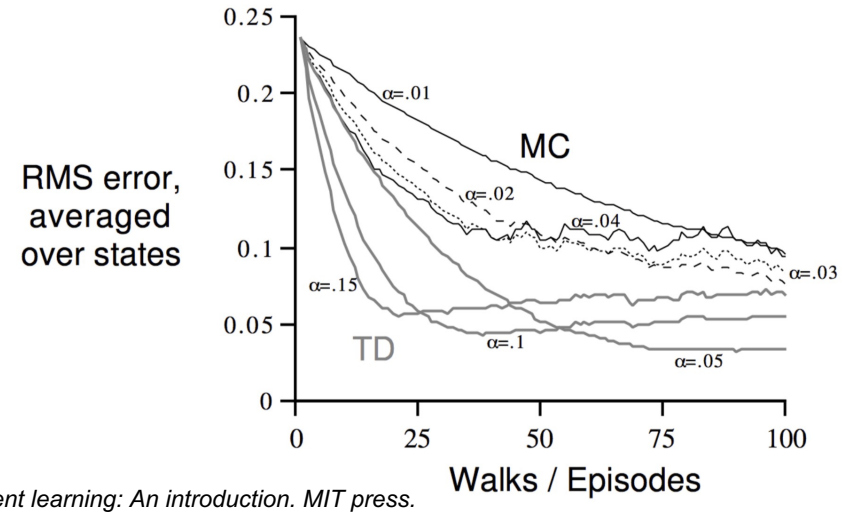
TD ($\alpha = 1$)

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Which one should I use? Does it make any difference?
- Example: Random Walk





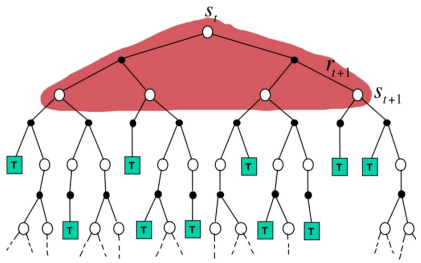
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Which one should I use? Does it make any difference?
  - TD can learn before (or even without) knowing the final outcome
    - after each step
    - incomplete sequences
    - continuing problems, very delayed or no return
  - MC only works for episodic problems (i.e., that terminate)
    - must wait until end of the episode

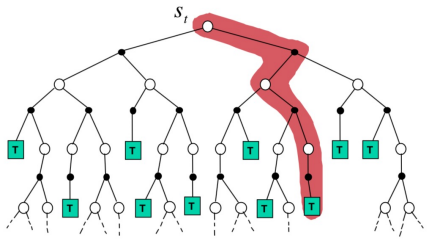
DP Backup

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



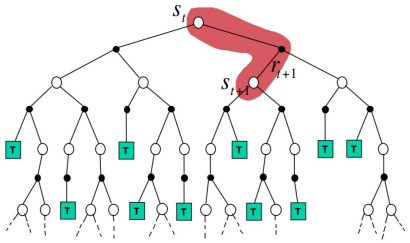
MC Backup

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



TD Backup

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



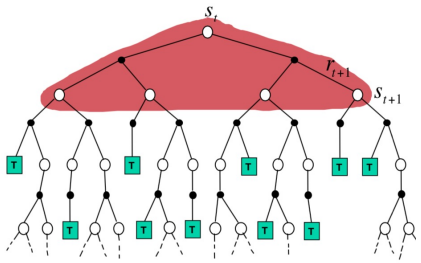
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Which one should I use? Does it make any difference?
  - Bias/Variance Trade-Off
  - MC has high variance, but zero bias
    - Good convergence (even with FA)
    - insensitive to initialization (no bootstrapping), simple to understand
  - TD has low variance, but some bias
    - TD(0) converges to $\pi_v(s)$ (be careful with FA: bias is a risk)
    - sensitive to initialization (because of the bootstrapping)
    - Usually more efficient in practice

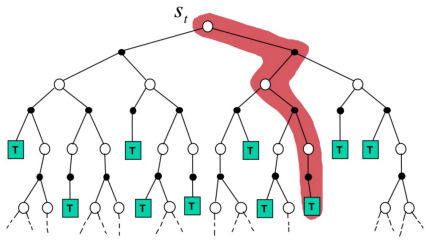
DP Backup

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



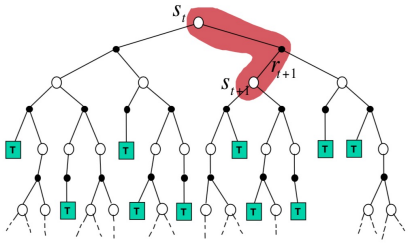
MC Backup

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



TD Backup

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



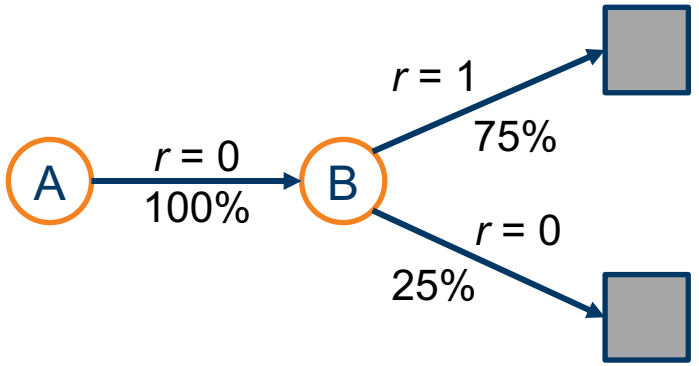
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Which one should I use? Does it make any difference?
- Example: You are the predictor!
  - Two states A, B; no discounting; 8 episodes of experience
  - keep iterating on experience (MC and TD until both of them converge):

```
A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0
```

  - What is $V(S = A)$ and $V(S = B)$?
    - MC: $V(A) = 0$ $V(B) = 0.75$
    - TD: $V(A) = 0.75$ $V(B) = 0.75$



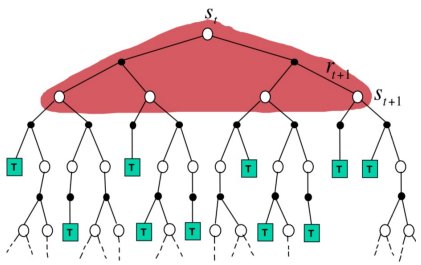
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Which one should I use? Does it make any difference?
  - TD exploits Markov property and is more efficient in Markov environments
  - MC is more efficient in non-Markov environments
  - TD usually converges faster than MC

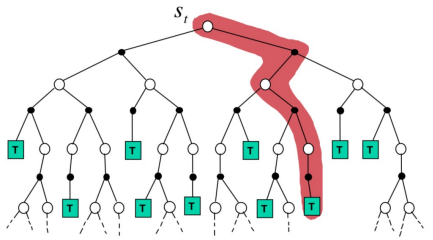
DP Backup

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



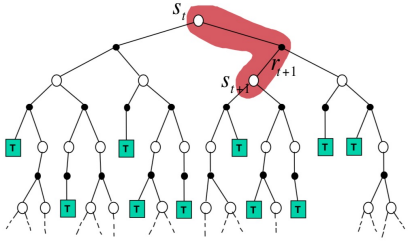
MC Backup

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



TD Backup

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



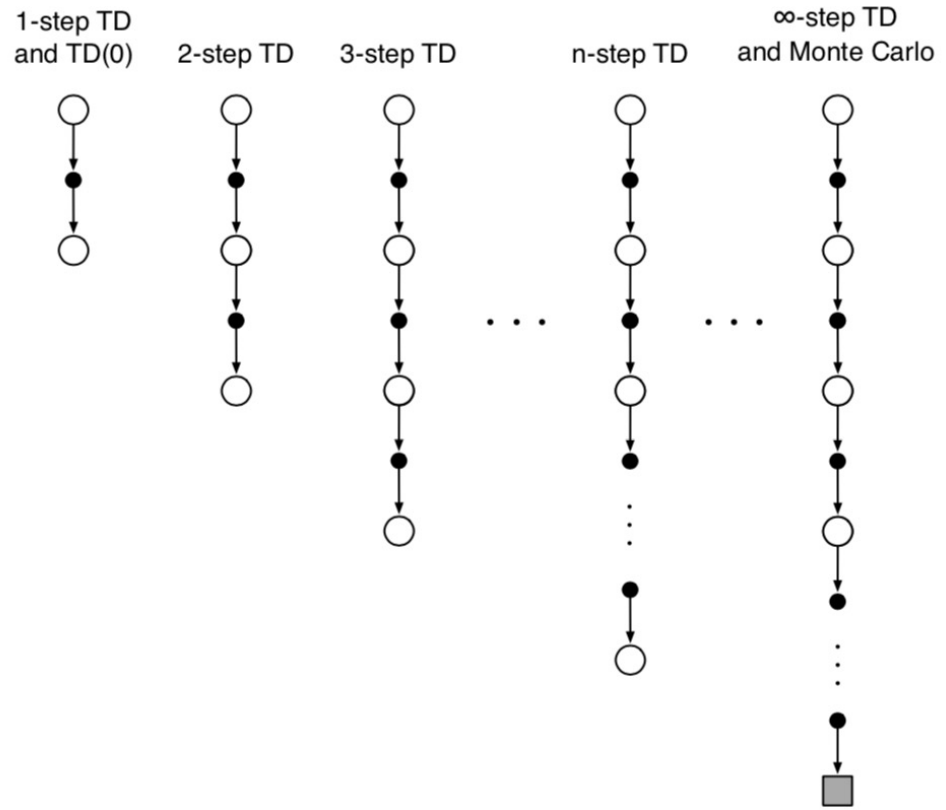
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

Hands-On:
https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

Monte Carlo and TD Methods

- Intermediate methods between MC and TD(0) exist
- They are based on n-step returns



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Intermediate methods between MC and TD(0) exist
- They are based on n-step returns

n-step TD for estimating $V \approx v_\pi$

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n + 1$

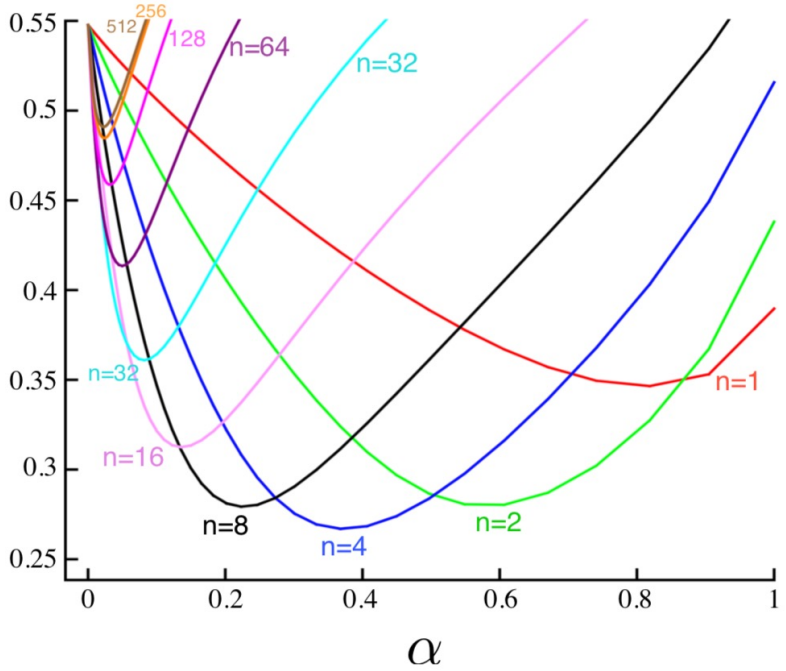
Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take an action according to $\pi(\cdot|S_t)$
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then $T \leftarrow t + 1$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

- Intermediate methods between MC and TD(0) exist
- They are based on n-step returns
- Unfortunately, their prediction accuracy is sensitive to the algorithm hyperparameters
- Example: Random Walk

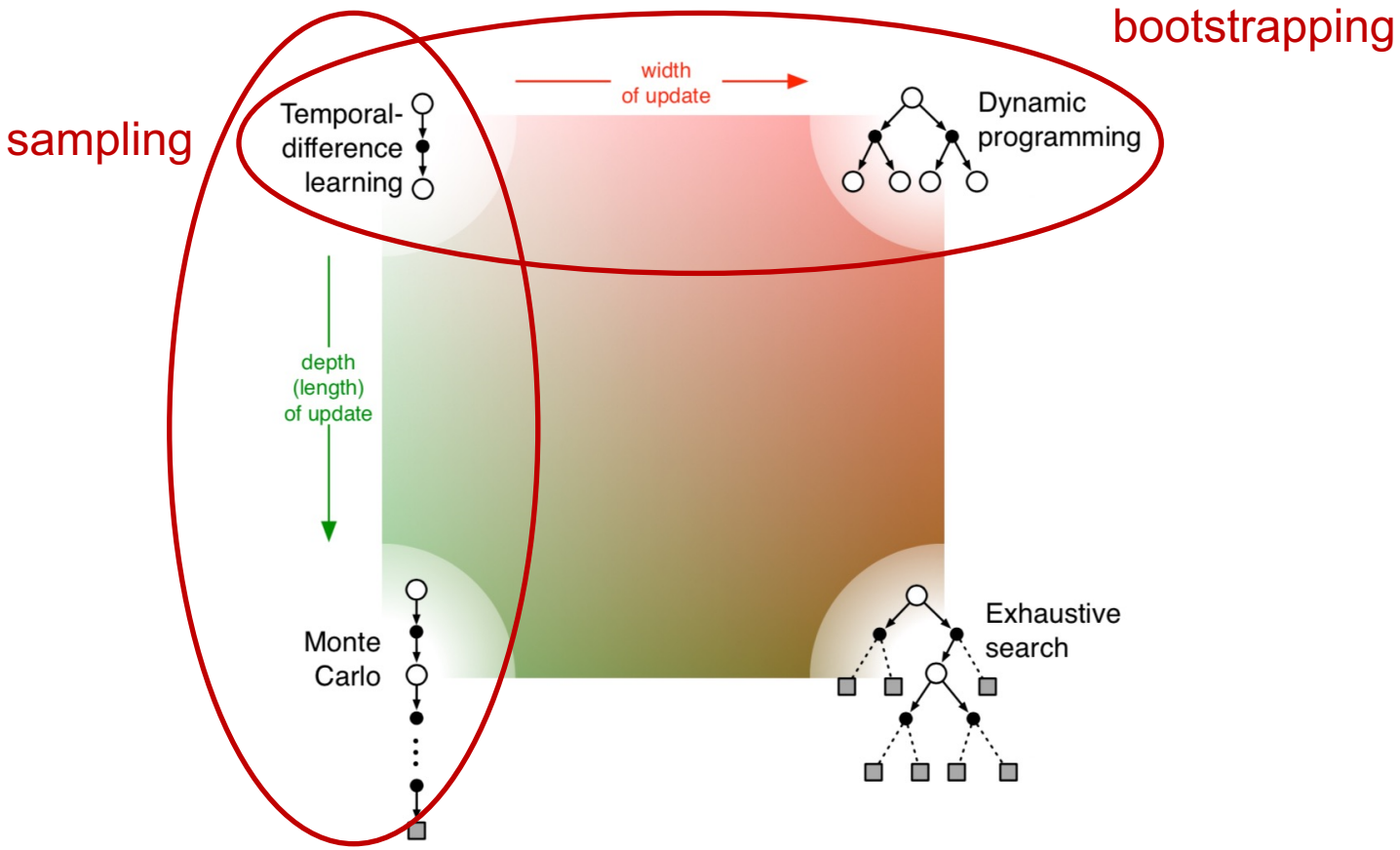
Average RMS error over 19 states and first 10 episodes



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Monte Carlo and TD Methods

A Unified View of Prediction Algorithms



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

TD – Remarks on Convergence Properties

- There is a lot of work that studied the convergence of TD:
- Convergence and optimality of (linear) TD methods under batch training (no online learning):
  - Richard S. Sutton: Learning to predict by the Methods of Temporal Differences. Machine Learning 3:9-44. 1988.
- Build on [Sutton1988] and proofs convergence of TD(0) and extends Watkin's Q-learning theorem (next video):
  - Peter Dayan: The Convergence of TD(λ) for General λ. Machine Learning 8:341-362. 1992.
- Further studies in the context of Q-Learning and SARSA (next video):
  - Tommi Jaakkola, Michael Jordan, Stainder Singh: On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. Technical report. 1994.
  - Francisco Melo: Convergence of Q-Learning: A Simple Proof. Technical report.
    (it has only 4 pages – so feel free to have a look ☺)
  - Satinder Singh, Tommi Jakkola, Michael Littman, Csaba Szepesvari: Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. Machine Learning 39:287-308. 2000.