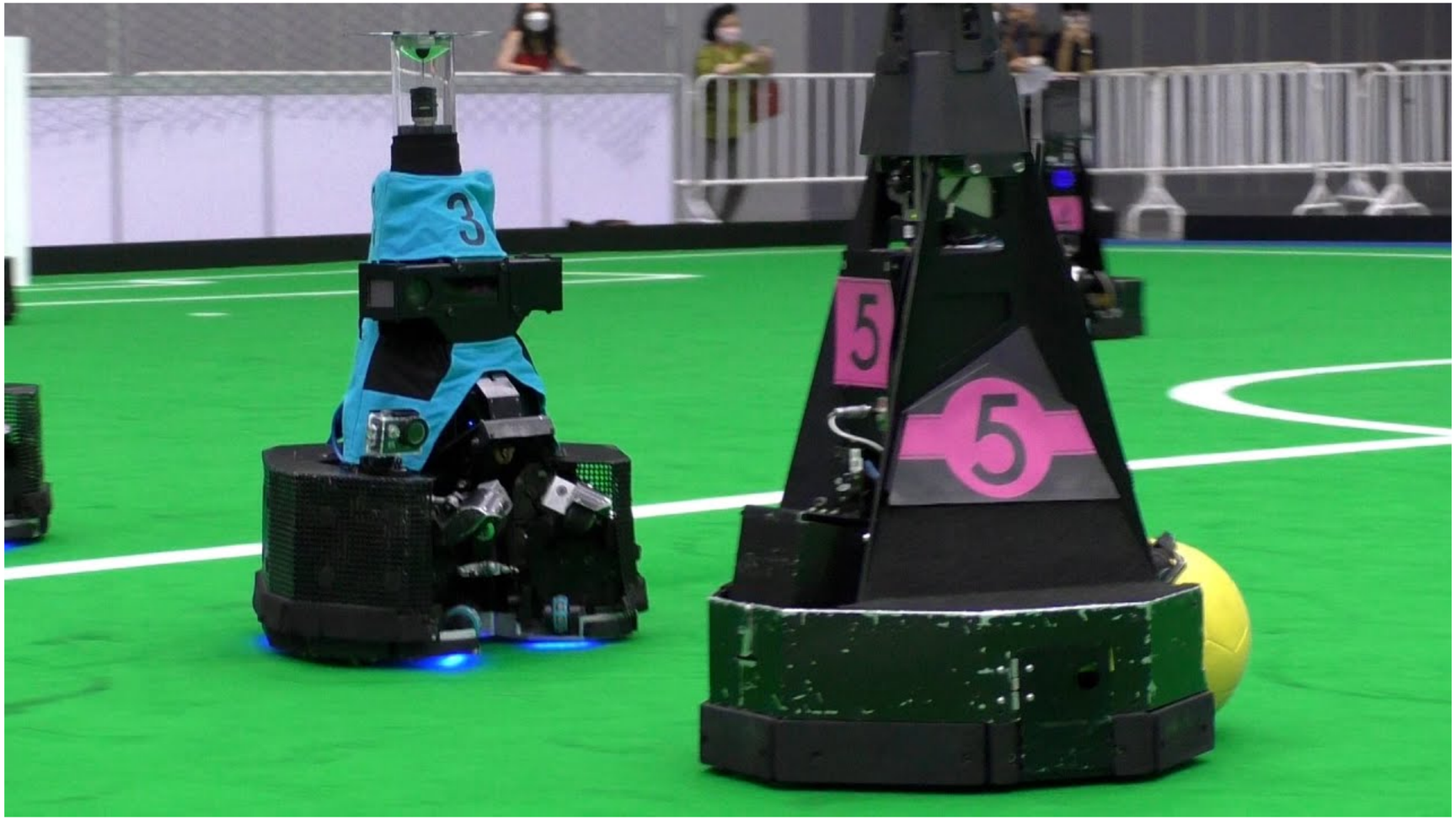
Temporal Difference Learning

CMPUT 655
Fall 2022


3
5
5

Perhaps the most significant algorithm in RL

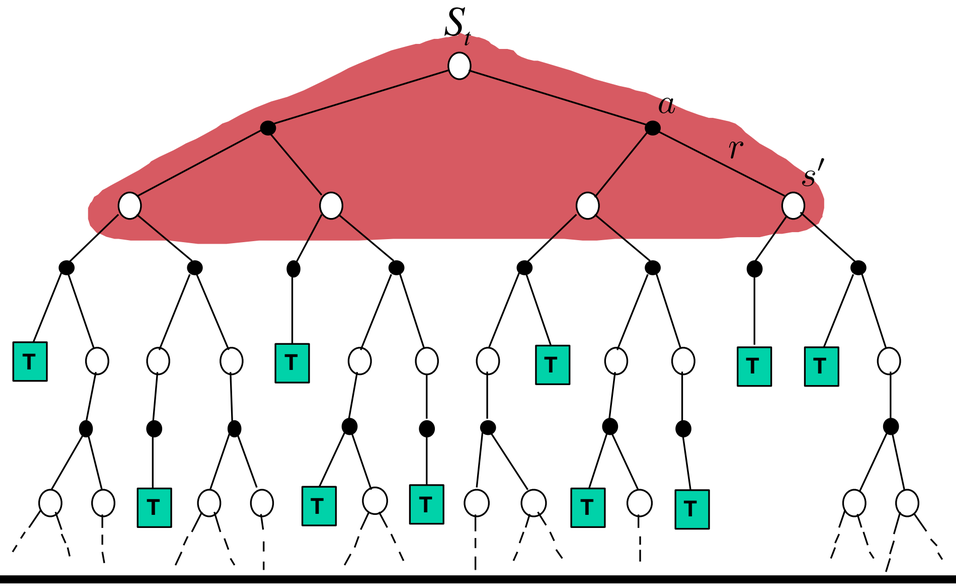
Tabular TD(0) for estimating v_π

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 $V(\mathit{terminal}) = 0$

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

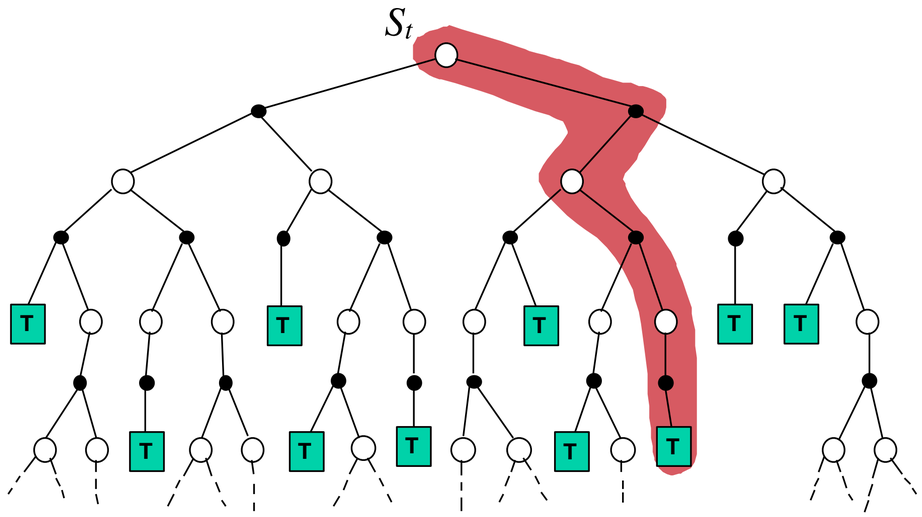
Dynamic programming

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



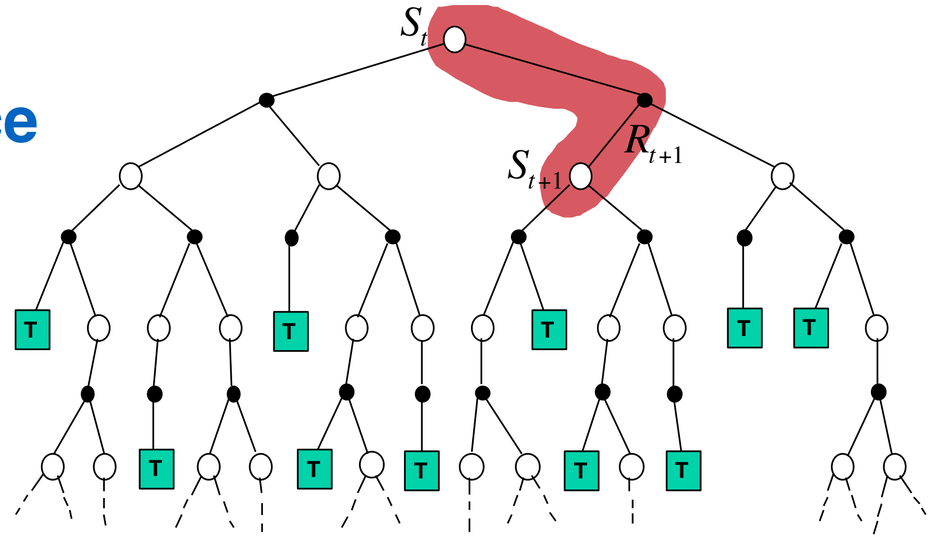
Simple Monte Carlo

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



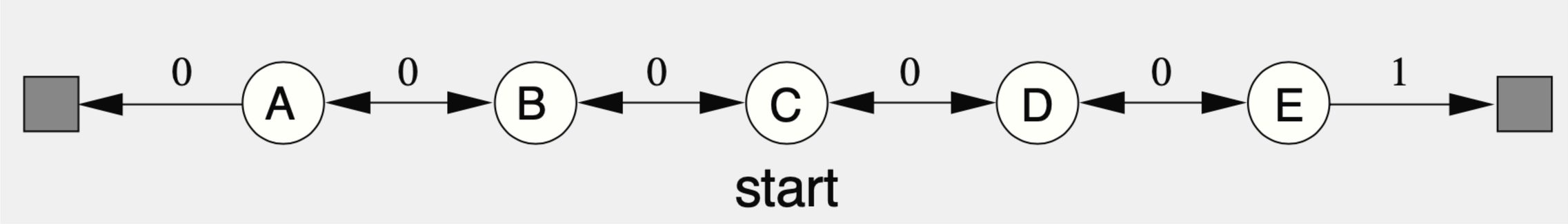
Temporal Difference Learning

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



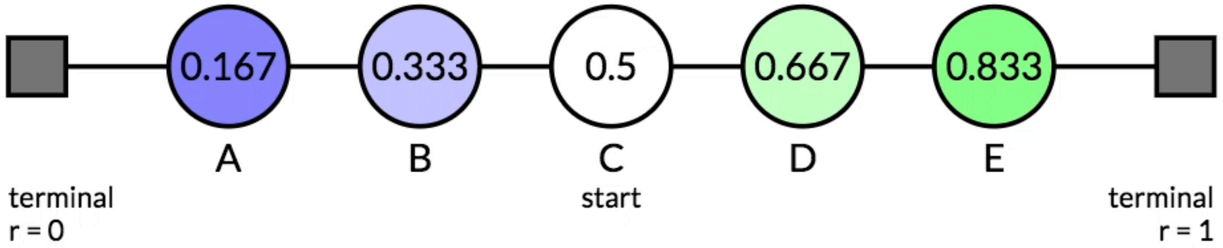
- *How can we understand the empirical advantages of TD over MC empirically? Let's look at some experimental results to better understand …*

A Random Walk problem

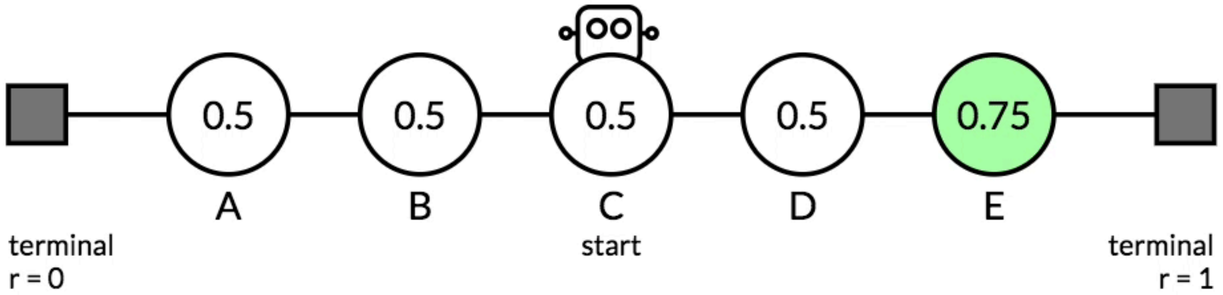


- Episodic; \gamma = 1.0
- Start in the centre
- Reward = 1 only on EXIT RIGHT
- What is the policy \pi?
- Goal: estimate v_\pi
  - What does v_\pi encode in this problem?

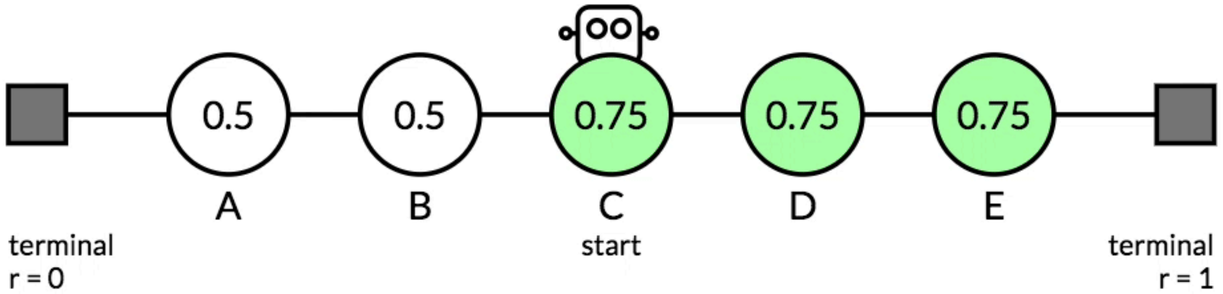
Target / Exact Values



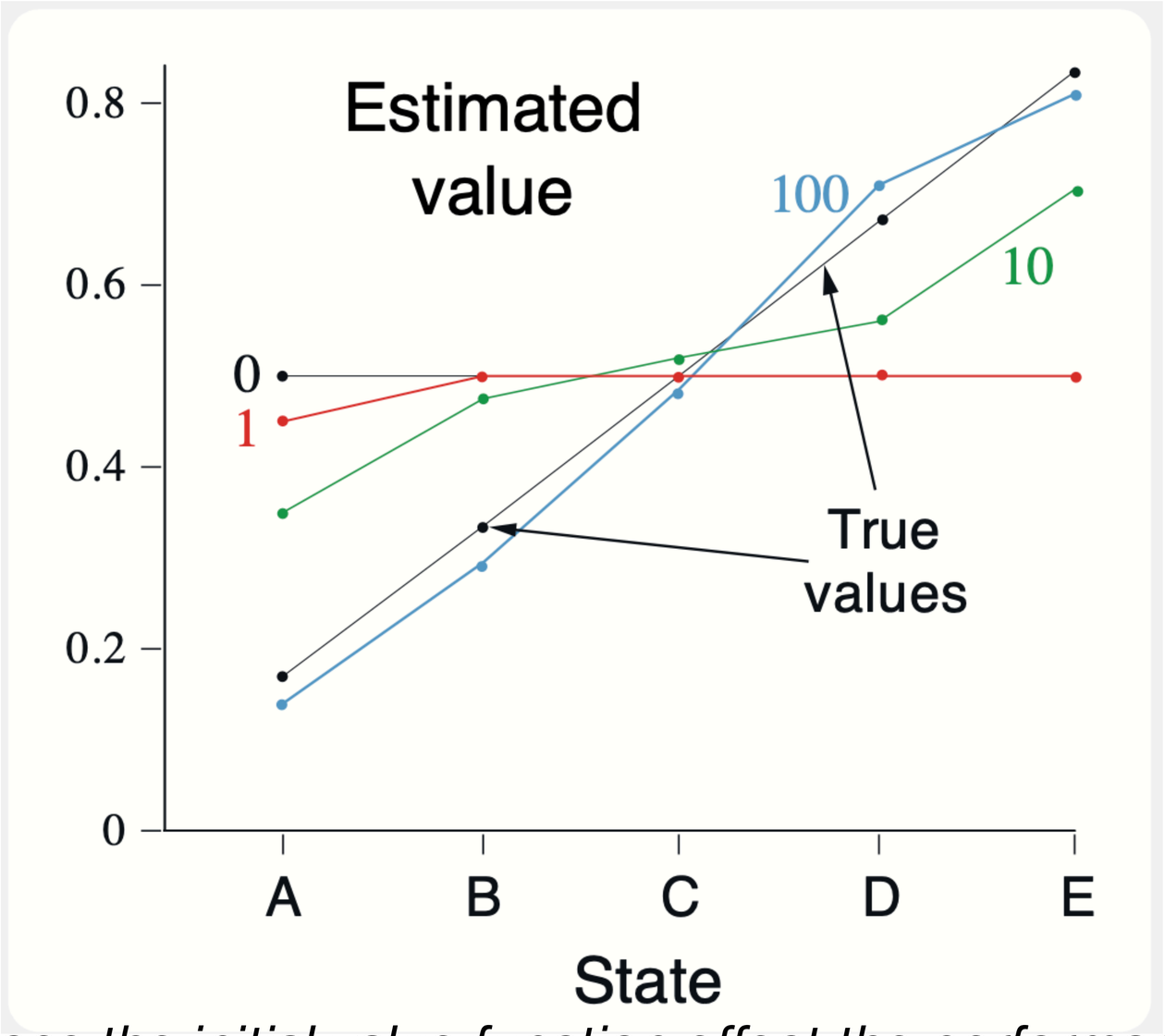
Updates using TD Learning



Updates using Monte Carlo

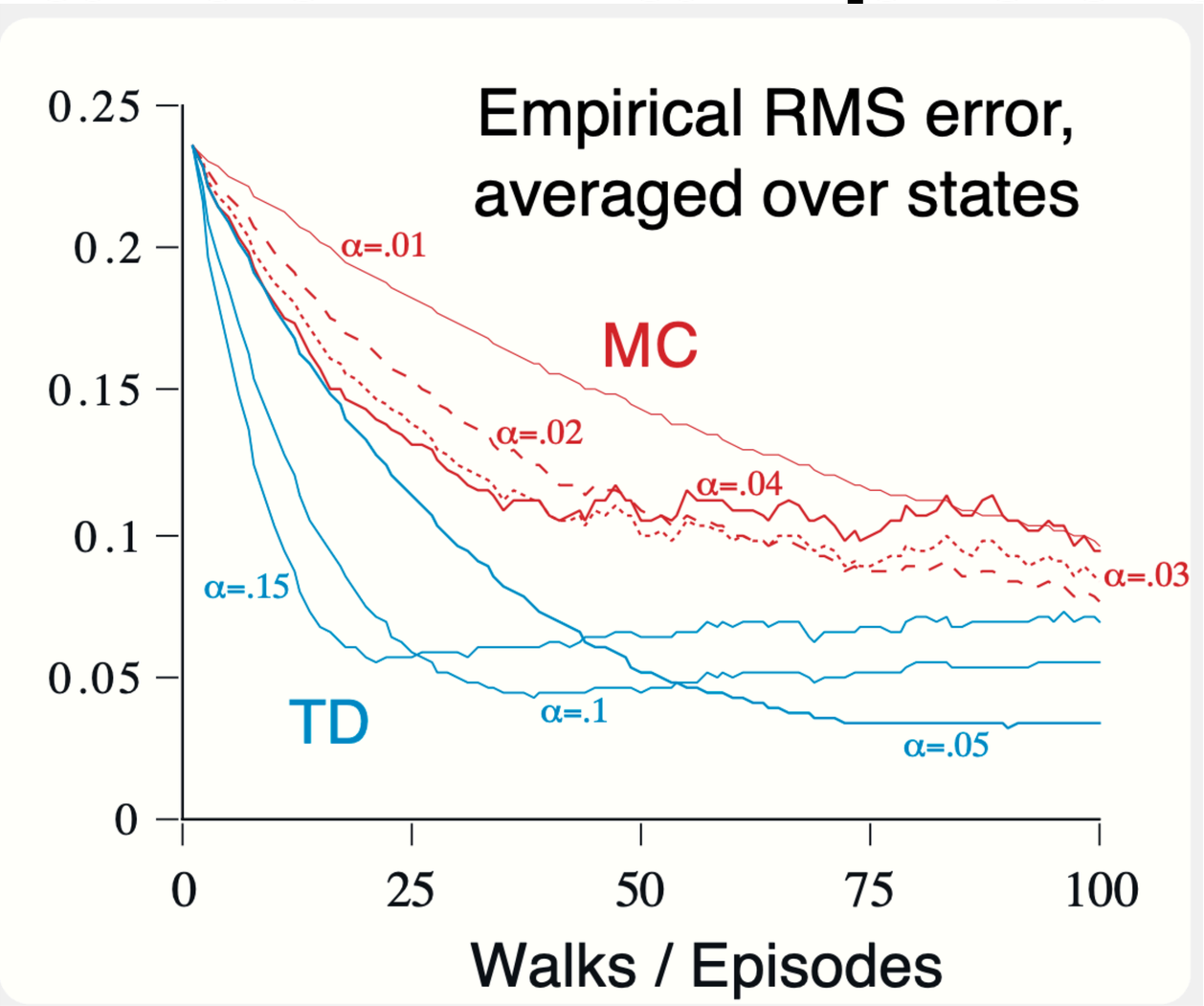


A Random Walk problem



- *In TD learning, does the initial value function effect the performance of the algorithm? Hint: look at the black line labelled ‘0’*

A Random Walk problem

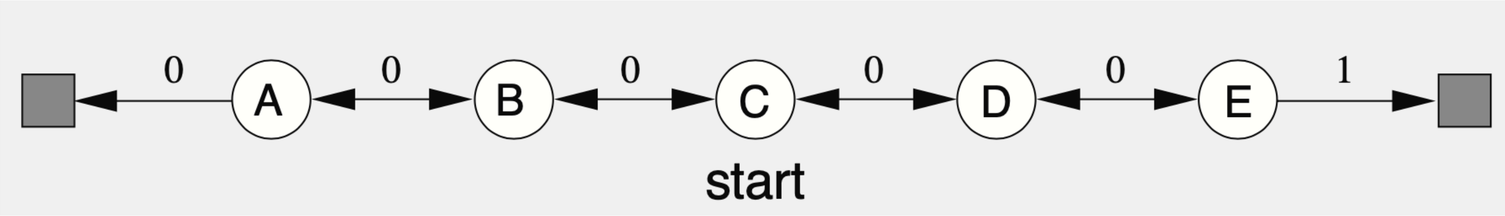


- *Why does the blue alpha=0.15 line go down fastest, but level off at a higher error?*

Back to our question

- *How can we understand the empirical advantages of TD over MC empirically?*
  - Let's think of the update targets for each:
    - MC: V(S_t) = V(S_t) + \alpha [**G_t** - V(S_t)]
    - TD: V(S_t) = V(S_t) + \alpha [**R_{t+1} + \gamma V(S_{t+1})** - V(S_t)]
    - Var[**R_{t+1} + \gamma V(S_{t+1})**] < Var [**G_t**]
- *When might MC be better empirically than TD?*

When might MC be better empirically than TD?



- Consider the Random Walk problem, estimate v_\pi, and \pi = always go right
- What is the return of the first episode? G = 1
- V(S_t) = V(S_t) + \alpha [**G_t** - V(S_t)]
  - *MC gets the value function correct after one episode! If alpha=1*
- What about TD? V(S_t) = V(S_t) + \alpha [**R_{t+1} + \gamma V(S_{t+1})** - V(S_t)]
  - *How many episodes would it take TD to get the value function correct?*
- The variance of the **one-step TD target** is not lower than the variance of the **return**
  - In this case TD is slowed down by the initially incorrect values in the target. Bootstrapping hurts!

Terminology Review

- In TD learning there are **no models, YES bootstrapping, YES learning during the episode**

- TD methods update the value estimates on a **step-by-step** basis. We **do not wait** until the end of an episode to update the values of each state.

- TD methods use **Bootstrapping**: using the estimate of the value in the next state to update the value in the current state: $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

  TD-error

- TD is a **sample update** method: update involves the value of single sample successor state

- An **expected update** requires the complete distribution over all possible next states

- TD and MC are sample update methods. Dynamic programming uses expected updates

Discussion posts

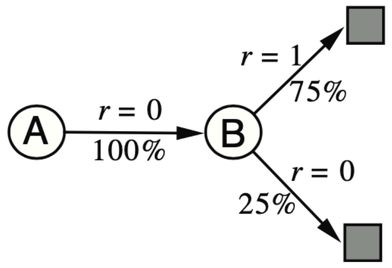
- ***MC converges to the optimal values V(s)***, the learning curve shows TD has a lower RMS error compared to MC

Example 6.4: You are the Predictor Place yourself now in the role of the predictor of returns for an unknown Markov reward process. Suppose you observe the following eight episodes:

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

This means that the first episode started in state A, transitioned to B with a reward of 0, and then terminated from B with a reward of 0. The other seven episodes were even shorter, starting from B and terminating immediately. Given this batch of data, what would you say are the optimal predictions, the best values for the estimates $V(\mathsf{A})$ and $V(\mathsf{B})$? Everyone would probably agree that the optimal value for $V(\mathsf{B})$ is $\frac{3}{4}$, because six out of the eight times in state B the process terminated immediately with a return of 1, and the other two times in B the process terminated immediately with a return of 0.

But what is the optimal value for the estimate $V(\mathsf{A})$ given this data? Here there are two reasonable answers. One is to observe that 100% of the times the process was in state A it traversed immediately to B (with a reward of 0); and because we have already decided that B has value $\frac{3}{4}$, therefore A must have value $\frac{3}{4}$ as well. One way of viewing this answer is that it is based on first modeling the Markov process, in this case as shown to the right, and then computing the correct estimates given the model, which indeed in this case gives $V(\mathsf{A}) = \frac{3}{4}$. This is also the answer that batch TD(0) gives.



- What are the connections between TD, neuroscience, and animal learning?
  - Dopamine (monkeys), classical conditioning (vs Rescolla wagner)
- Does TD work well online with big RNNs
  - Yep! Maybe DQN is not “online enough”?. There is work in that direction:
    - More online replay: https://arxiv.org/abs/1712.01275
    - Fixed target replay: capstone course of the RL MOOC
- How do we compare TD and MC?
  - Why is this more annoying than control?
  - Are their standard benchmarks? Soon an Atari prediction data set

- What are some practical advantages to (or examples of) choosing a MC method over a TD method like SARSA and Q-Learning?

- For real world problems, is it really that important to be able to learn incrementally? What are some advantages of incremental learning that make TD algorithms more preferable?

  - Maybe we are just not working on such problems yet…consider the driving home example

- TD is biased because the initial value estimates are wrong. Is that a problem? Is bias always bad?

  - When could MC be better?

- TD's theoretical support is mostly in the tabular case and linear function approximation case. Is that odd? Should we be worried?

  - Consider the universal function approximation results for RNNs

- TD has been used for modelling animal data in classical conditioning. But there are some things the TD model, does not…model well
  - Rapid reacquisition, for example
- Why do we compare TD and MC in the batch setting? Do people ever actually use batch TD?
  - You bet!
- What theoretical support do we have for the convergence of TD?
  - Convergence in expectation, wp1, rates of convergence, off-policy, non-linear…
    - See chapter on TD with function approximation
  - Remember: the historical remarks section at the end of the textbook!

- What kinds of predictions can we make with TD?
  - Ones about the accumulation of some signal over time
  - Is this too limited?
- Could we take some notion of confidence to adjust the amount of bootstrapping?
  - Good idea!
- Can we mix TD and MC?
  - Yep! N-step TD, TD(lambda). What does the zero in TD(0) mean anyway?
- What is the deadly triad?
  - Bootstrapping, off-policy, and function approximation

- What is the deadly triad?
  - Bootstrapping, off-policy, and function approximation

Eligibility traces

- Chapter 12, plus some extra stuff!
- Like n-step methods, but better!

Eligibility traces

- A mechanism that allow TD, Sarsa and Q-learning to learn more efficiently
- A way to move **smoothly** between TD(0) updates and Monte Carlo updates (using full returns)
  - useful when we don't have the Markov property
  - allows us to do Monte Carlo-like updates even when the task is continuing or episodes are very long
  - methods in-between are often better in practice
- Allows TD methods to do multi-step updates
- Much easier to implement than n-step methods

1-step prediction

- Consider the update target of contant-α MC
  - $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T$
  - uses the full return, all future rewards until termination
- Consider the update target of TD(0)
  - $G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
  - it's only based the next reward and $\gamma V(S_{t+1})$ is a stand-in for the rest of the return $\gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$
  - this allows us to avoid waiting, we can update on every step

n-returns prediction

- We do something in between
  - use more than one reward, but less than all rewards until termination
- Use two rewards and then the value function:
  - $G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$
  - now $\gamma^2 V(S_{t+2})$ takes the place of: $\gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots$
  - we call this a 2-step return
- We could do 3-step, 4-step,….. n-steps
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- We call these n-step TD methods; TD(0) is a one-step TD method

n-step returns

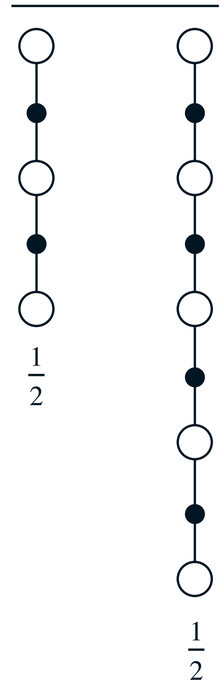
- All n-step returns are approximations of the **full return** G_t
  - truncated after n-steps and corrected for the remaining missing terms using $V(S_{t+n})$
- Monte Carlo methods are a special case being an ∞-step return

TD updates with n-step returns

- We can do these updates **online**, during the episode as soon as the information becomes available:
  - we wait n-1 steps
- n-step methods
  - they work better than TD(0)
  - BUT the algorithms are more complex

Complex backups

- Another valid option is averaging n-step returns
- For example we could average a 2-step and 4-step return
  - $0.5\ G_{t:t+2} + 0.5\ G_{t:t+4}$
- In fact, we can average any set of returns this way, even infinite set of returns
  - as long as the weighting sums to 1
  - still guaranteed to converge to correct predictions
- The average of simple backups is called a **complex backup**, TD(λ) uses a complex backup



λ-returns

- The TD(λ) algorithm averages n-step returns in a particular way
- Each n-step return is weighted proportional to λ^{n-1}, $\lambda \in [0,1]$
  - Normalized by (1- λ)
- For the continuing case the λ-return is:

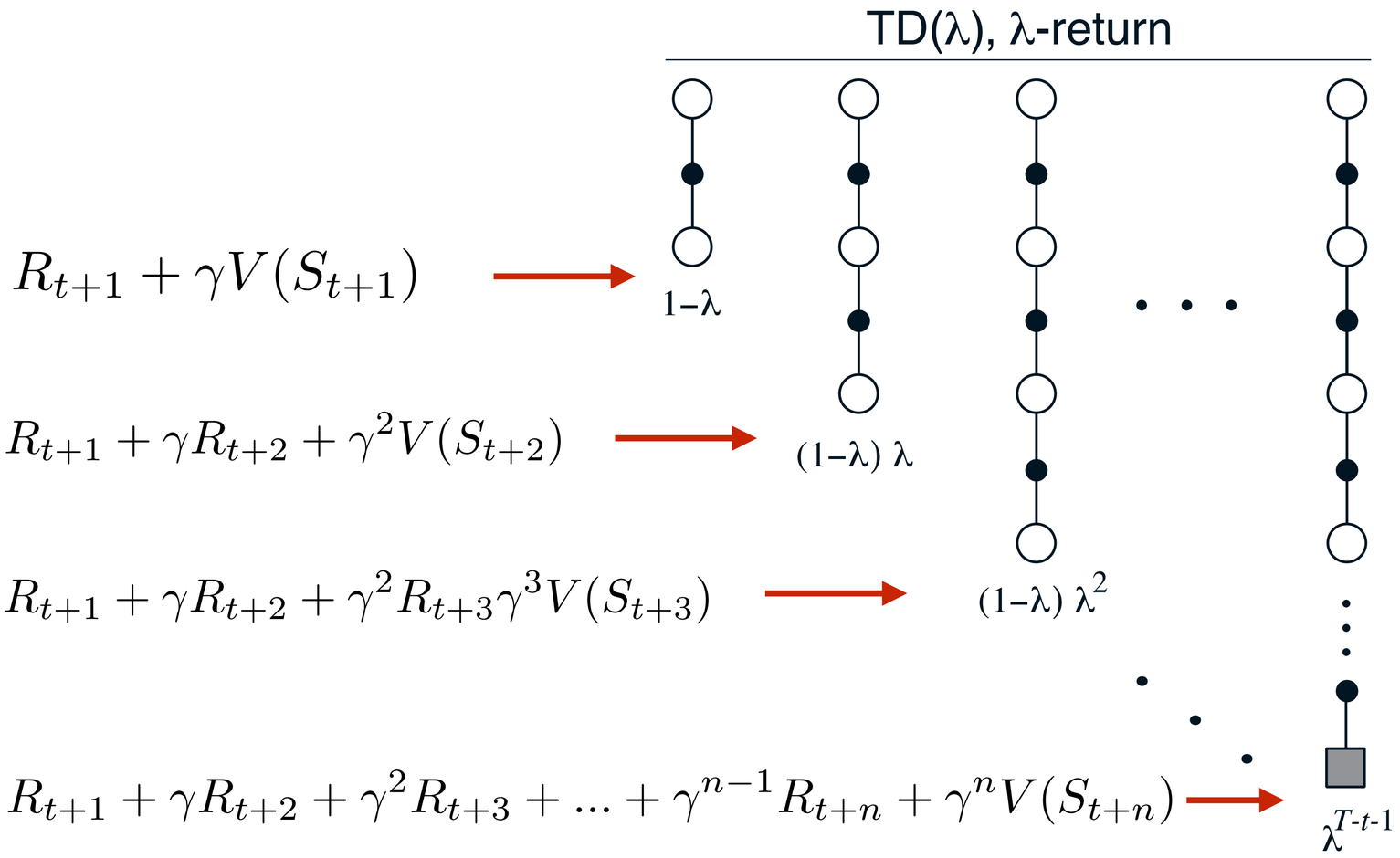
$$G_t^\lambda \doteq (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$$

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1}R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1}), \quad 0 \le t \le T-n$$

- When λ=1 $G^\lambda_t = G_t$, the full return, Monte Carlo
- When λ=0 $G^\lambda_t = G_{t:t+1}$, the 1-step return, **TD(0), hence the name!**

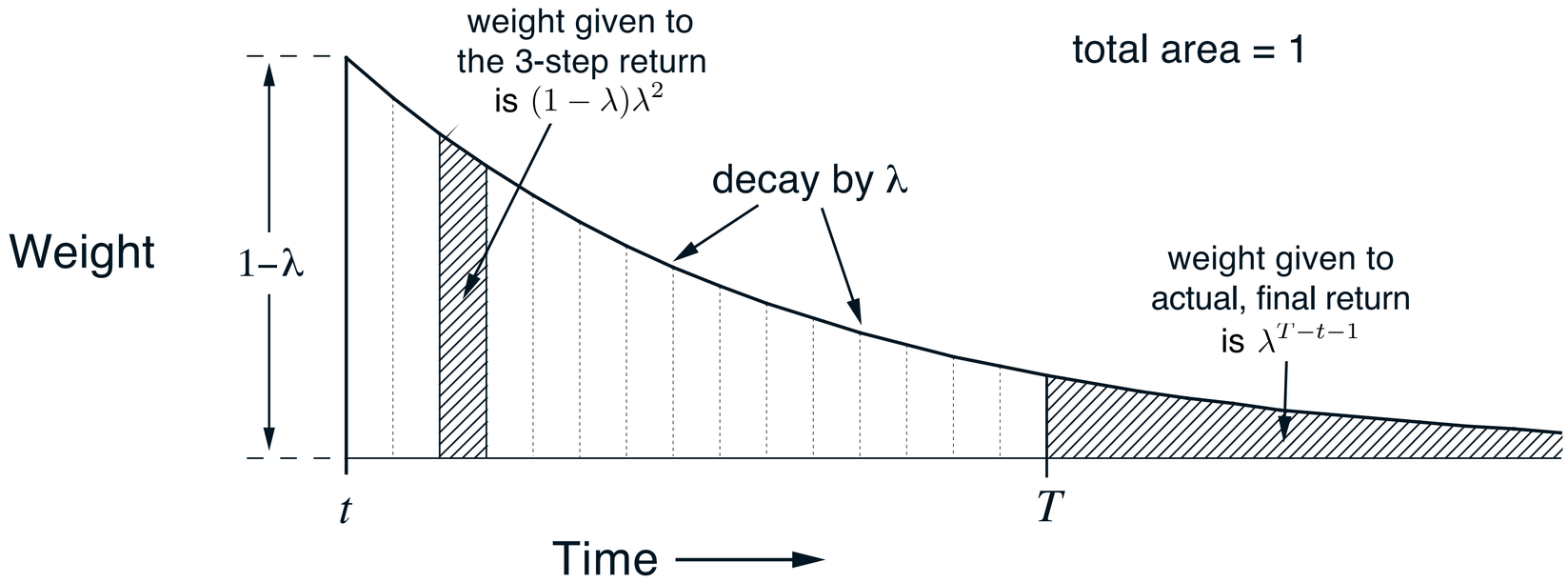
Episodic case λ-returns

$$G_t^\lambda \quad = \quad (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} \quad + \quad \lambda^{T-t-1} G_t$$

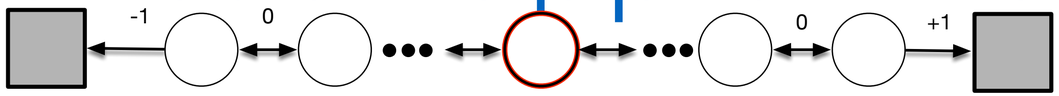


λ-returns

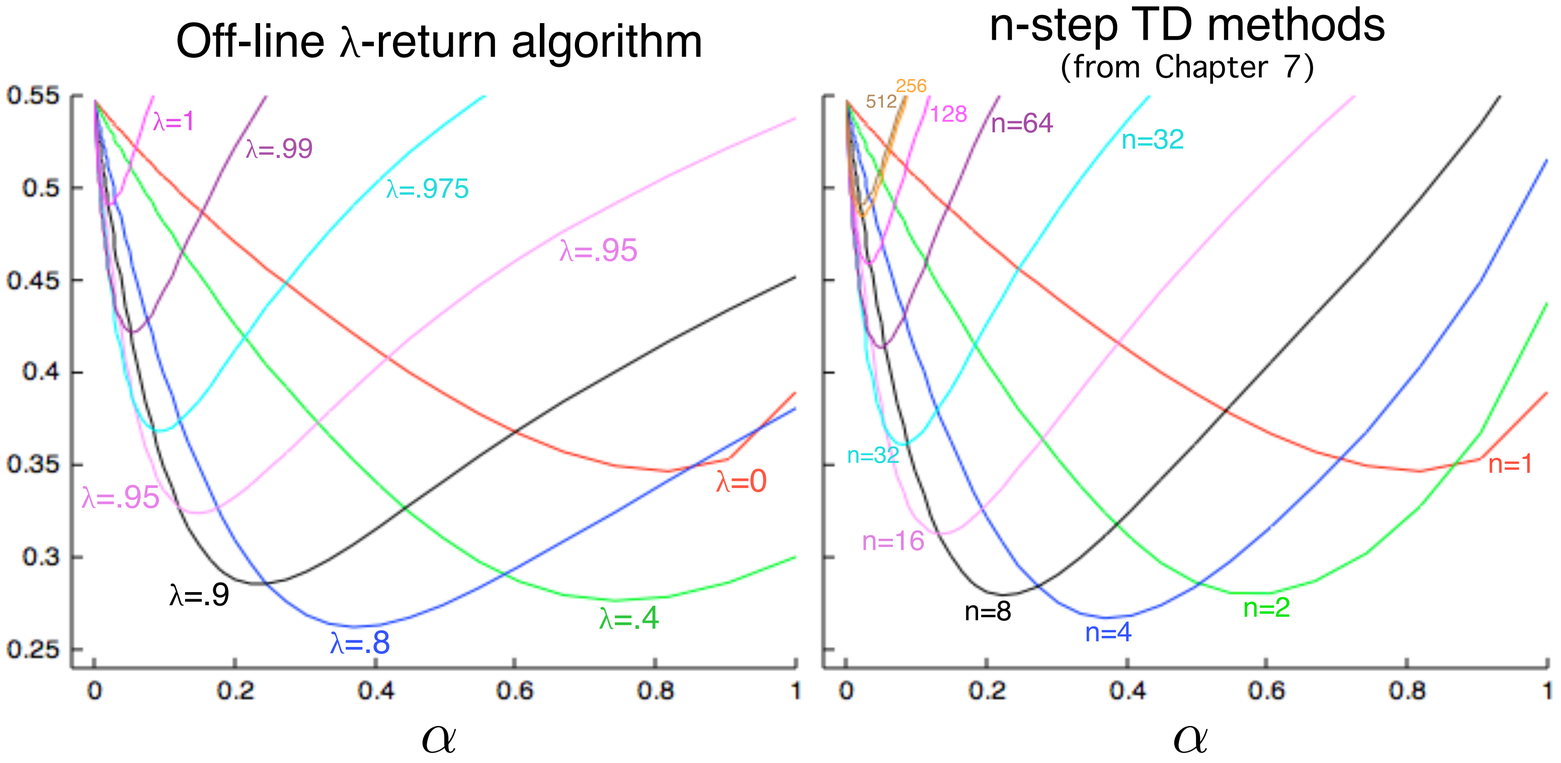
$$G_t^\lambda \quad = \quad (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} \quad + \quad \lambda^{T-t-1} G_t$$



19 State chain (no function approximation)

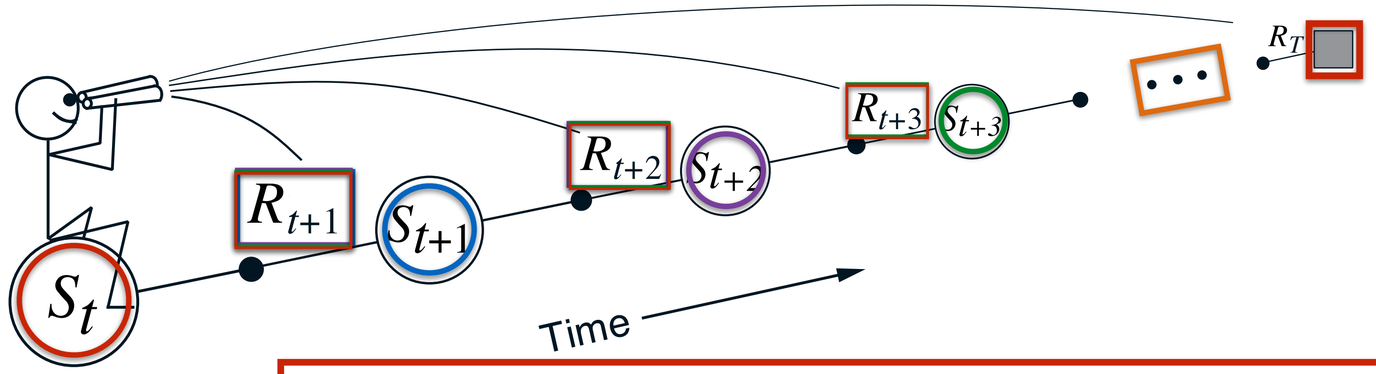


RMS error at the end of the episode over the first 10 episodes



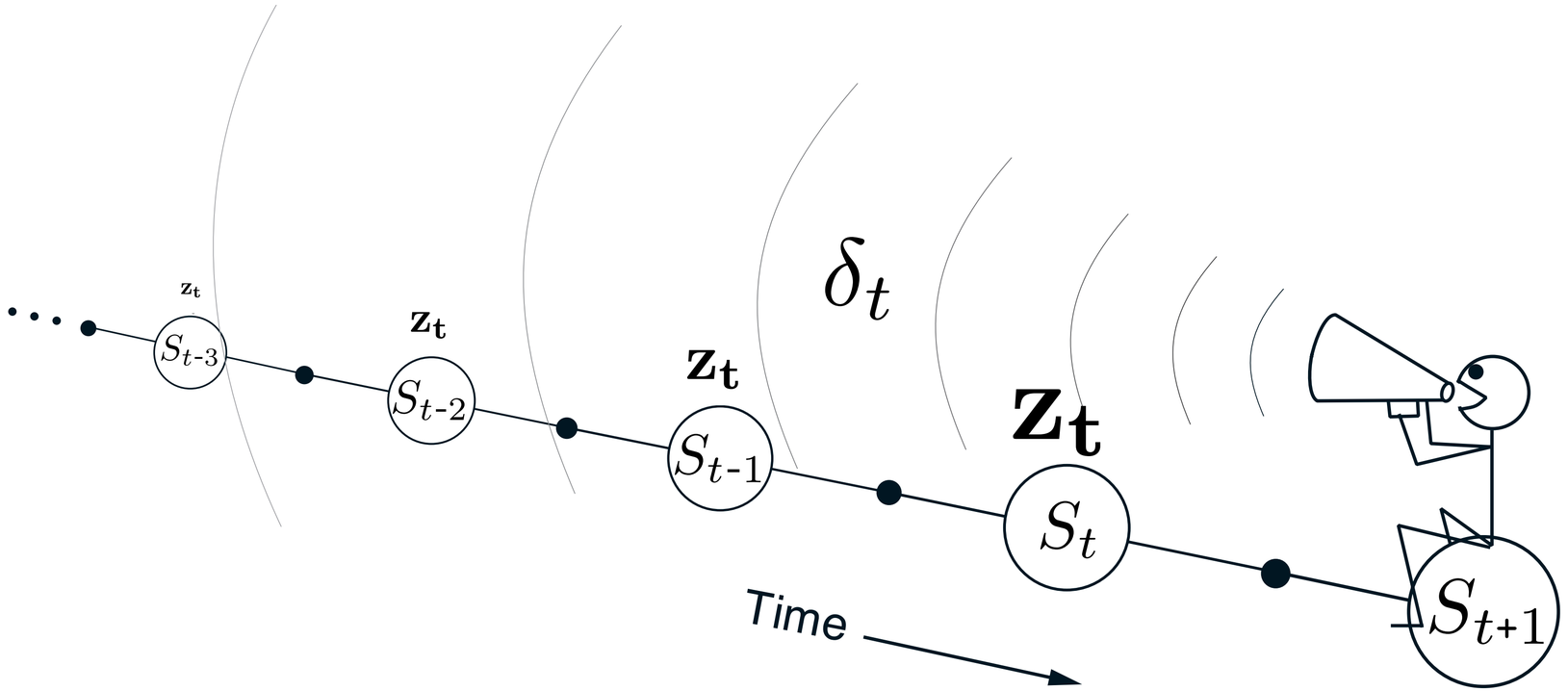
The forward view

- The λ-return algorithm uses what is called the *forward view*
- From each state we visit, we look forward in time, to all future rewards



$$G_t^\lambda = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t$$

The backward view looks back to the recently visited states (marked by eligibility traces)



- Shout the TD error backwards
- The traces fade with temporal distance by $\gamma\lambda$

The backward view

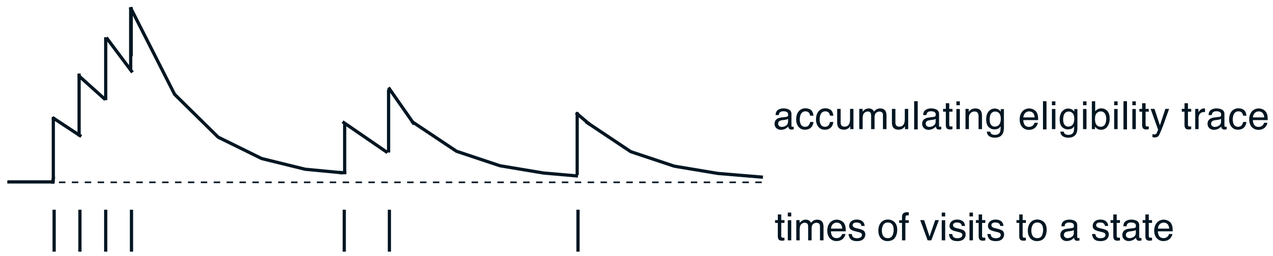
- The forward view, is *theoretical*, it's not directly implementable
  - at each step it requires knowledge of what will happen on **future** steps
- The backward view provides a **mechanism** for **approximating** the forward view
- We need a memory variable called the *eligibility trace*

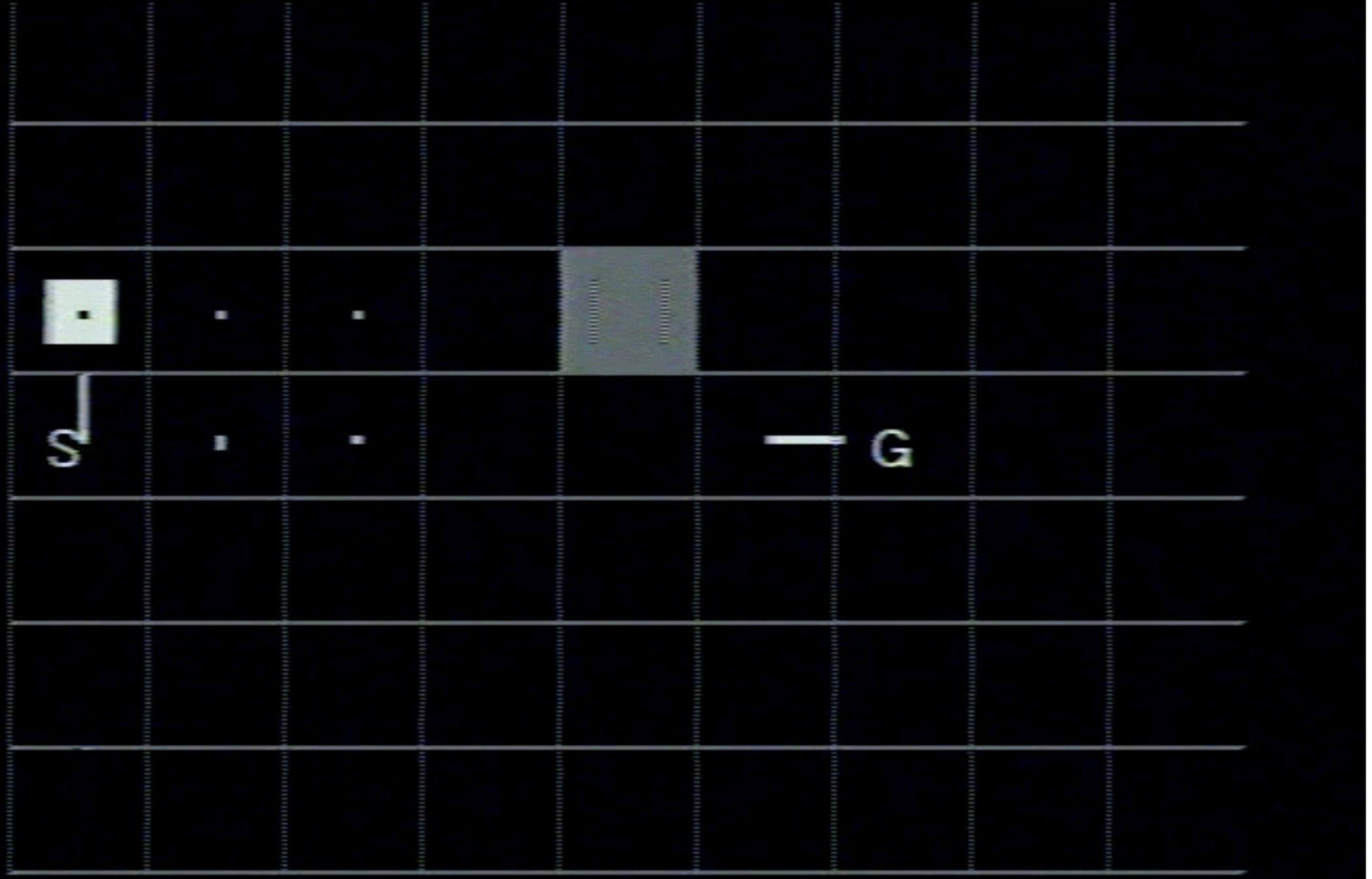
The TD(λ) algorithm

- One of the oldest, best known, and most successful algorithms in RL
- Three distinctive advantages over off-line λ-return algorithm:
  - updates weight vector (value func) on every step, not just at the end of episodes
  - computation equally distributed (not just at end)
  - can be applied to continuing problems

Eligibility traces

- The eligibility traces keep a record of which states have been visited recently
- The traces indicates the degree to which a state is eligible for a learning update





Here we are marking state-action pairs with a replacing eligibility trace