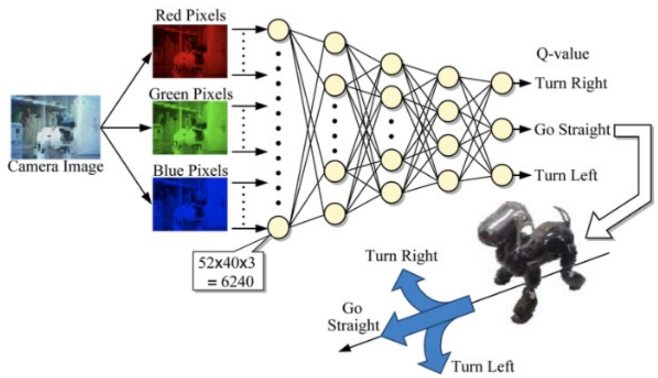
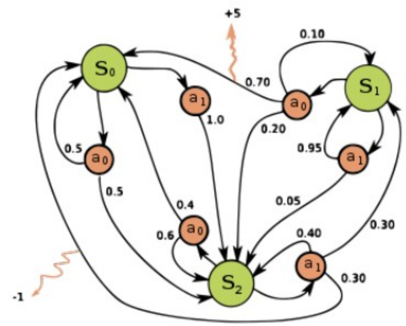
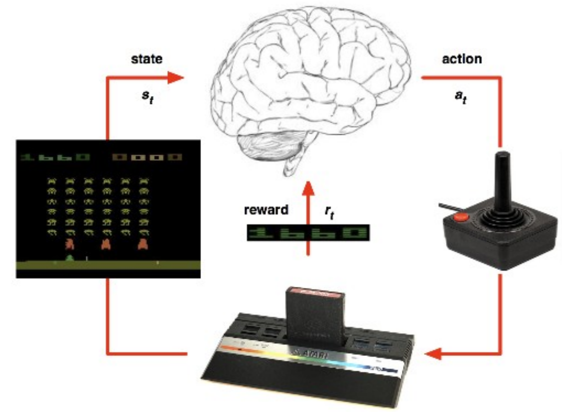
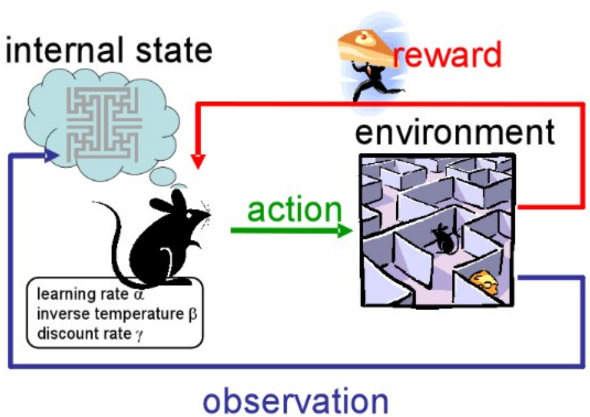
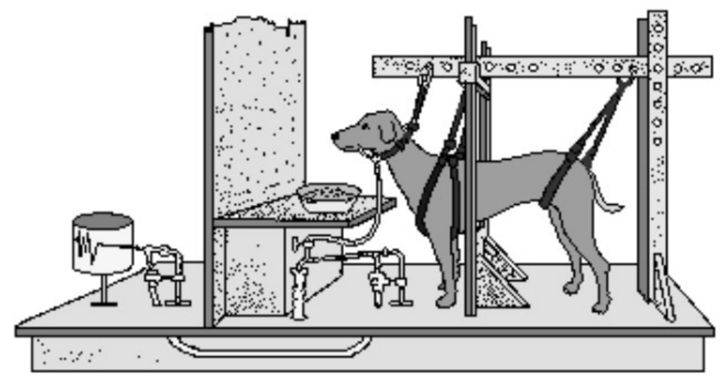
COMP 138: Reinforcement Learning











Instructor: Jivko Sinapov
Webpage: https://www.eecs.tufts.edu/~jsinapov/teaching/comp150_RL_Fall2021/

Today

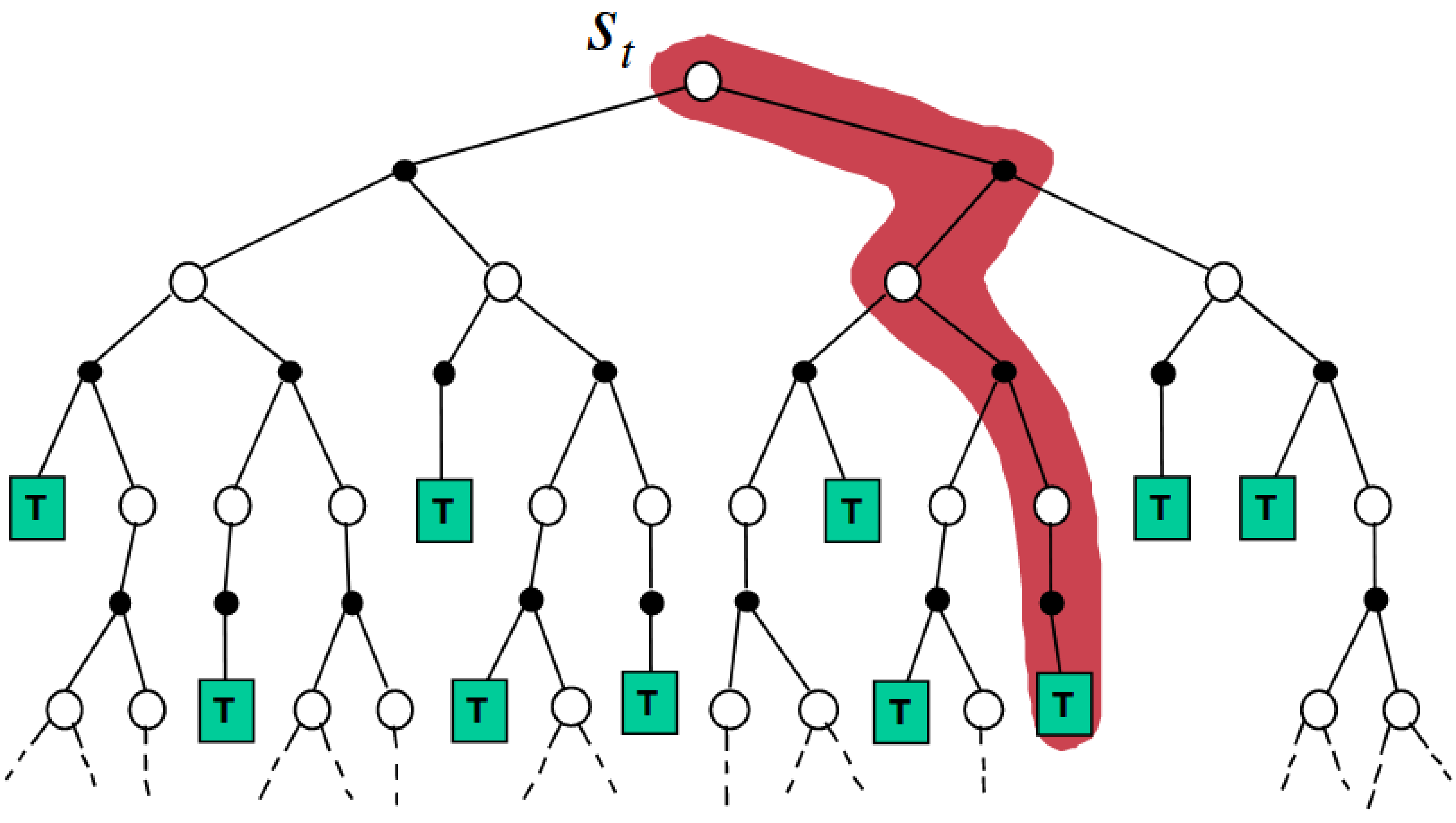
- Introduction to Eligibility Traces
- Project Breakout

Reading Assignment

- Chapter 13: Policy Gradient
- A research article of your choice

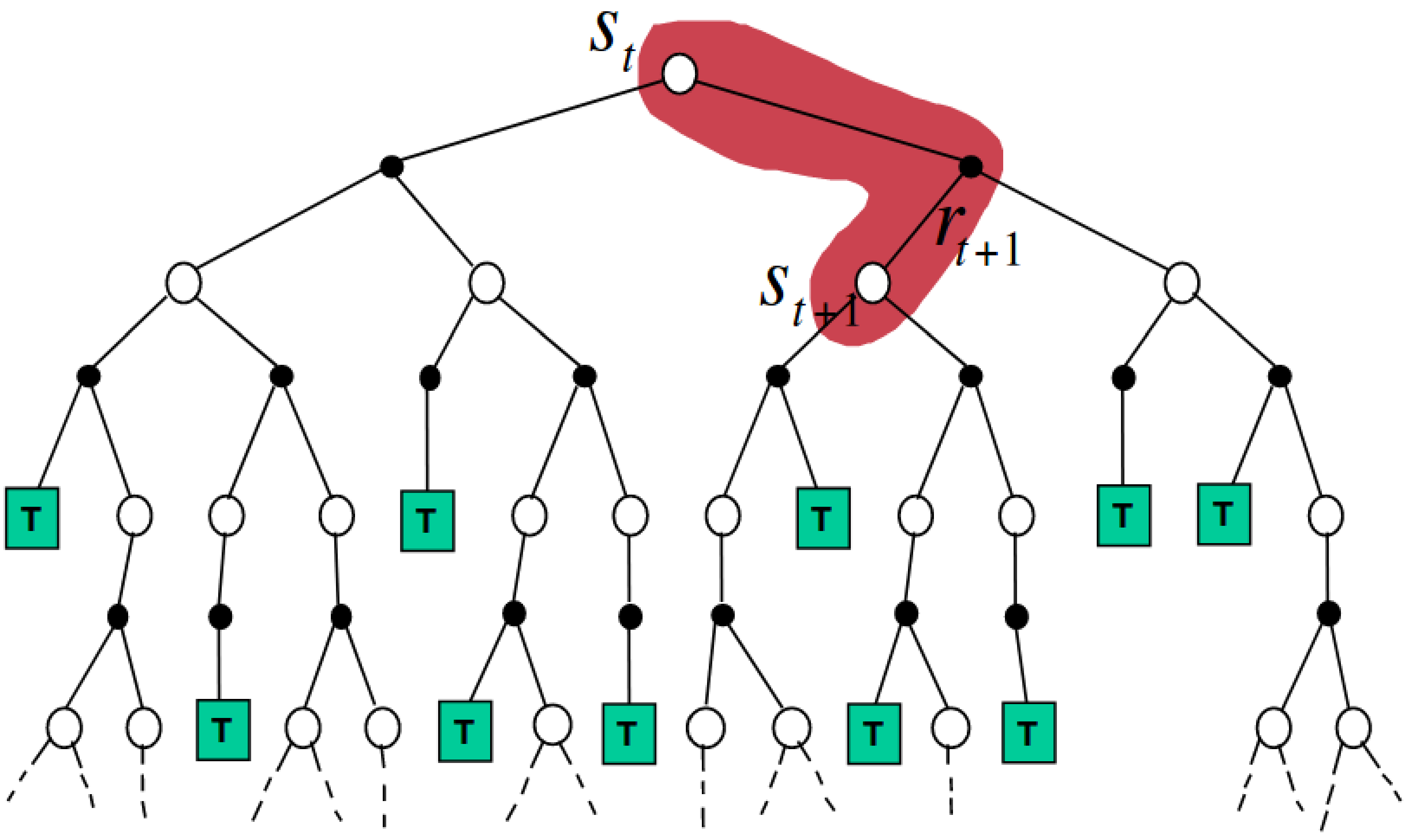
MC Backup

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



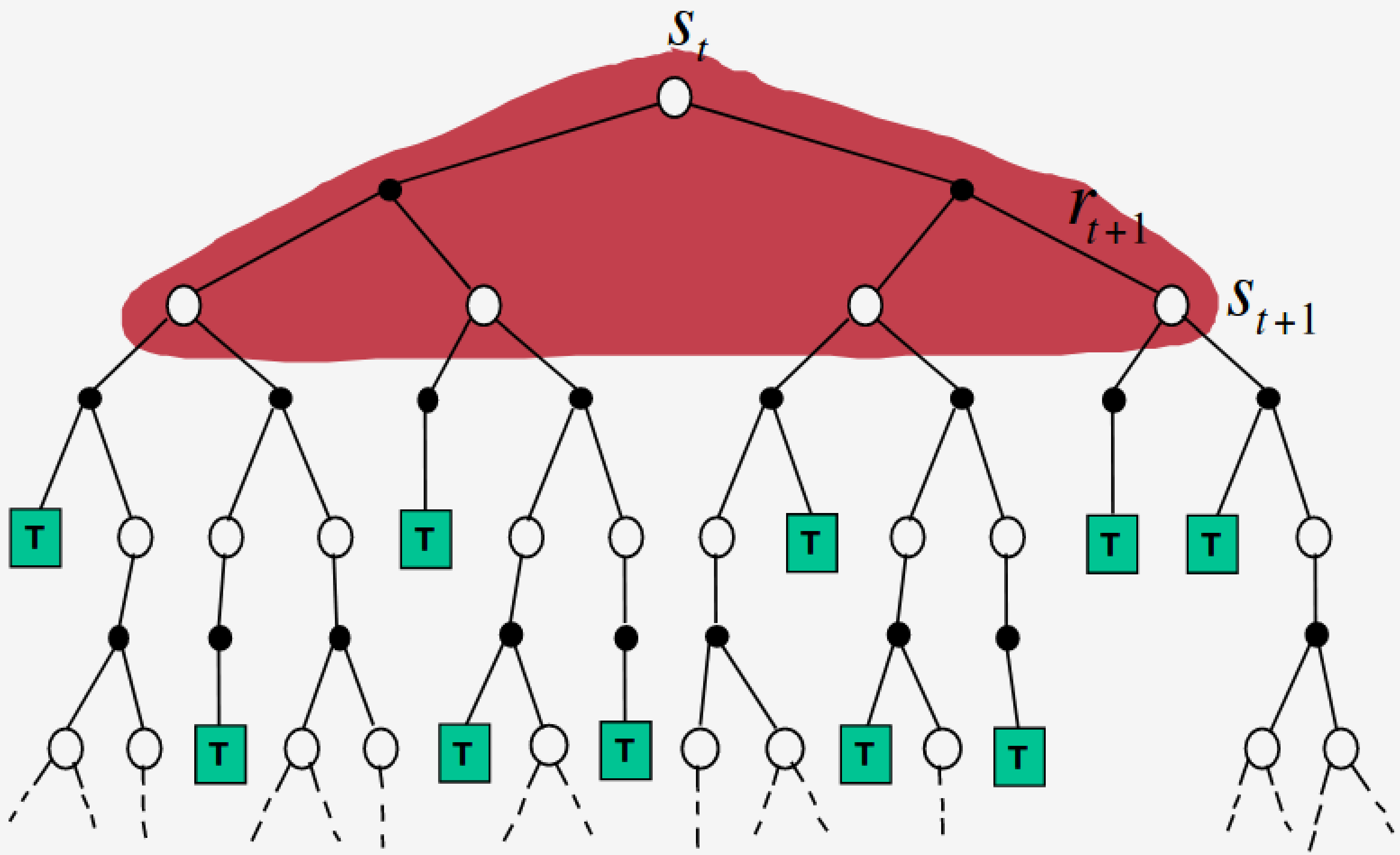
Temporal Difference Backup

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



Dynamic Programming Backup

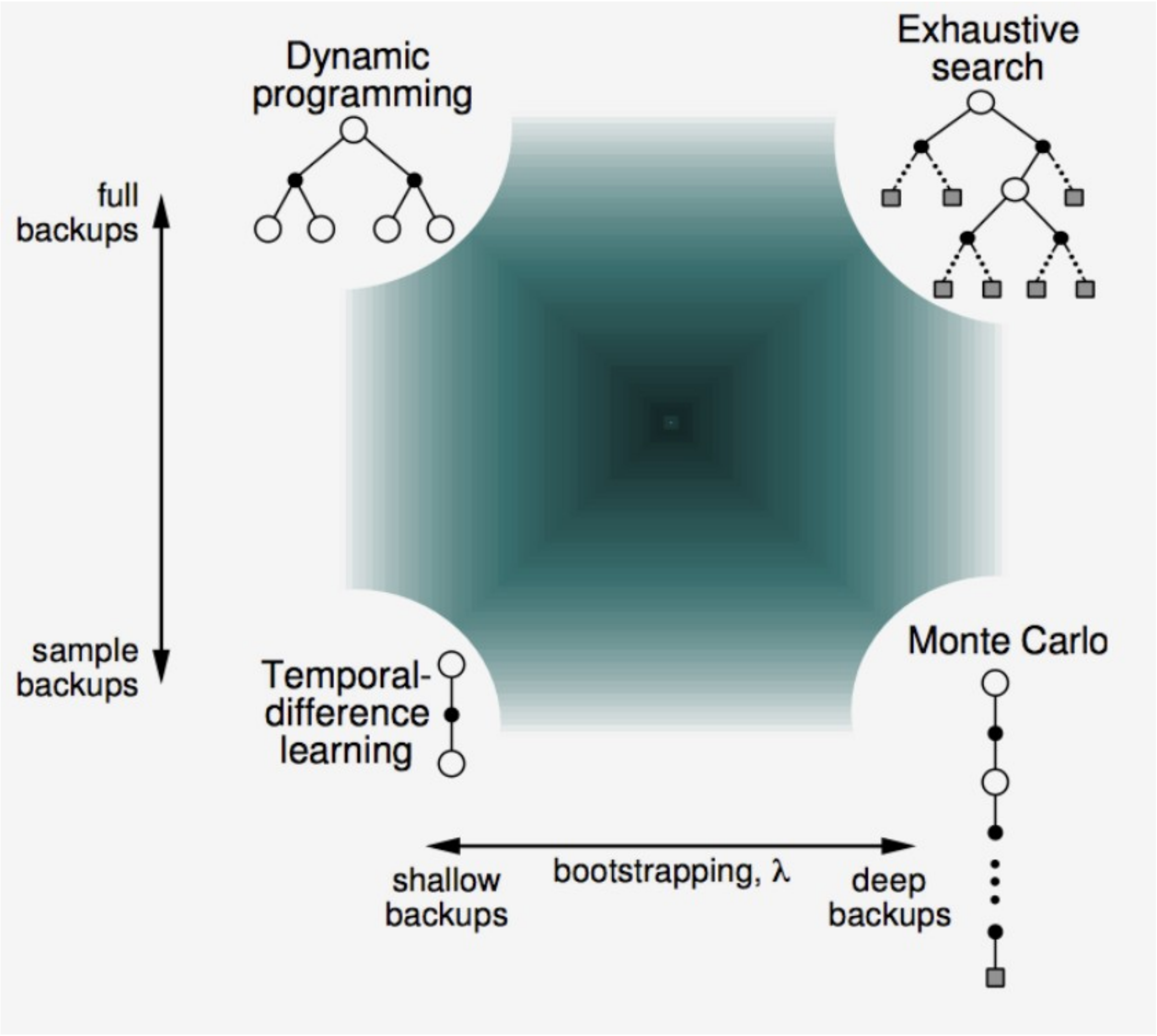
$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



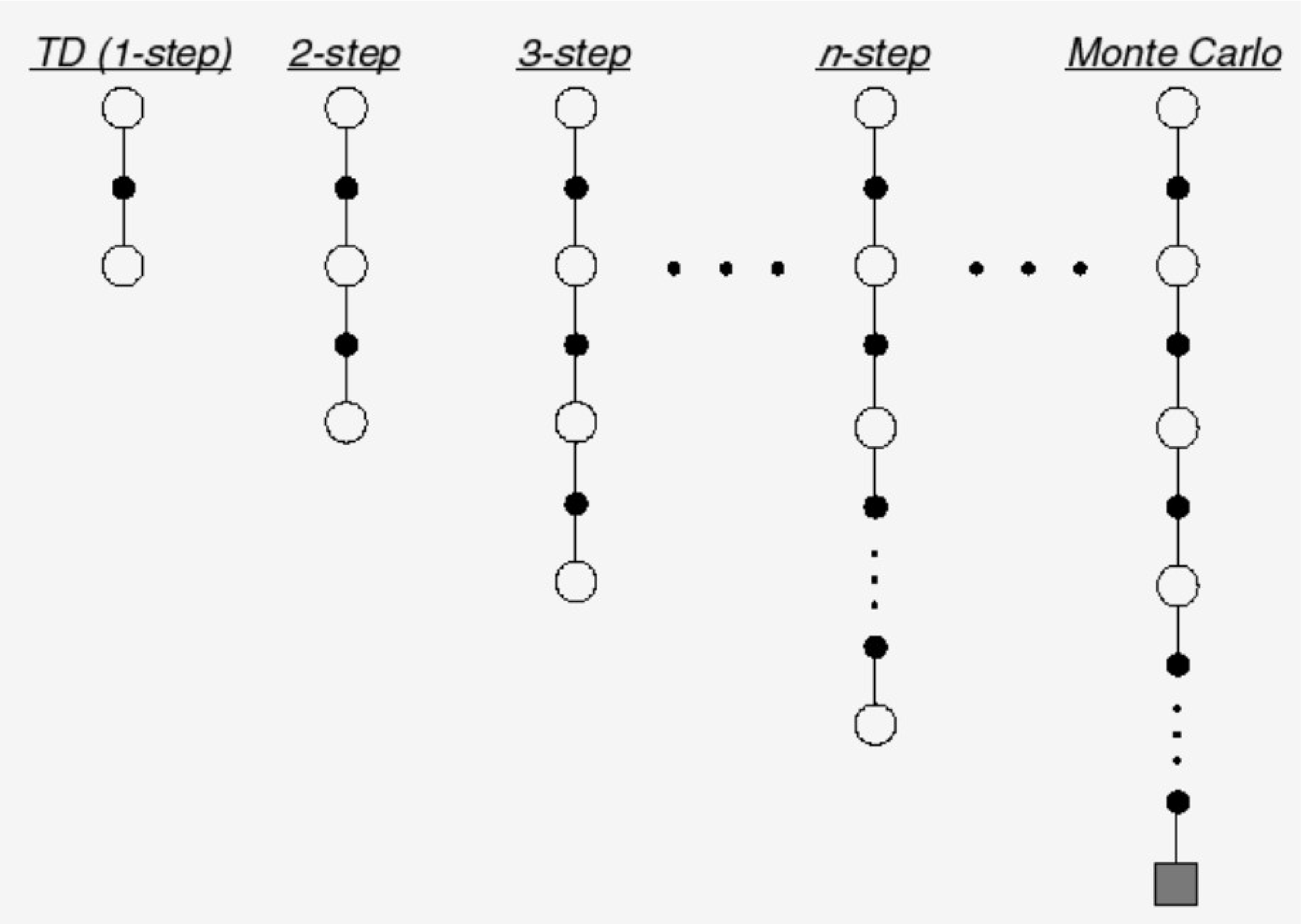
Bootstrapping vs Sampling

- Which of these methods bootstraps? Which samples?

Unified View of RL



n-Step Prediction



n-Step Return

n-Step Return Definition:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

n-Step Returns for different n:

$$\begin{array}{lll} n = 1 & (TD) & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\ n = 2 & & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\ \vdots & & \vdots \\ n = \infty & (MC) & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T \end{array}$$

TD Learning using n-Step Returns:

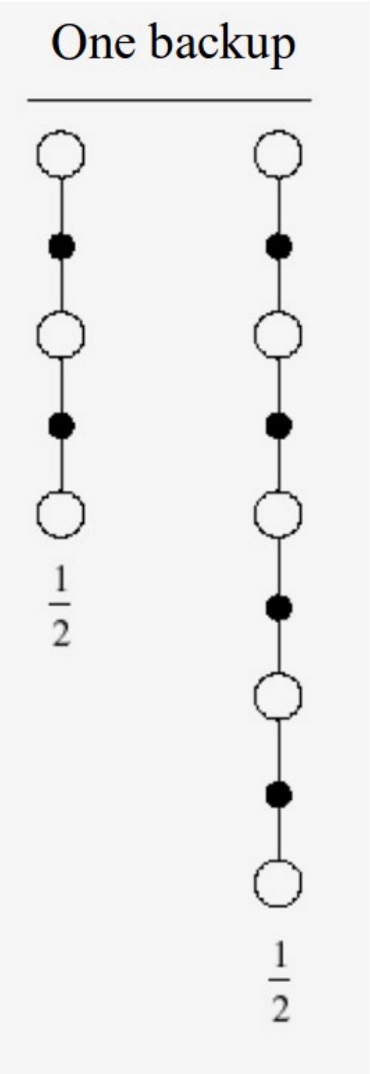
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

Averaging n-Step Returns

- n-Step returns can be averaged
- For example, average of 2-step and 4-step return is:

$$\frac{1}{2}G^{(2)} + \frac{1}{2}G^{(4)}$$

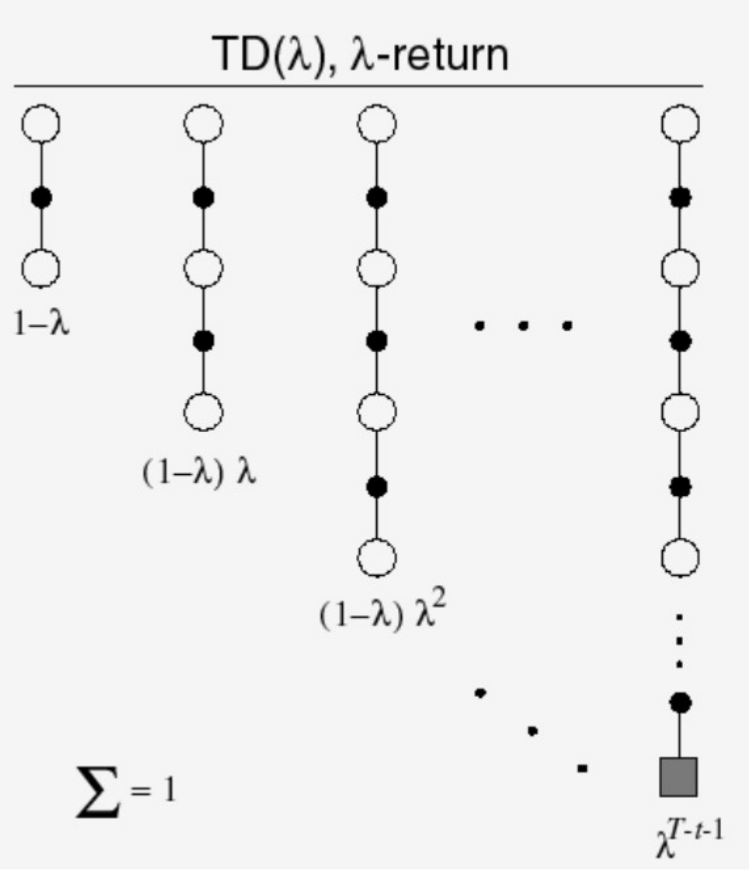
- Can we efficiently combine information from from all time steps?



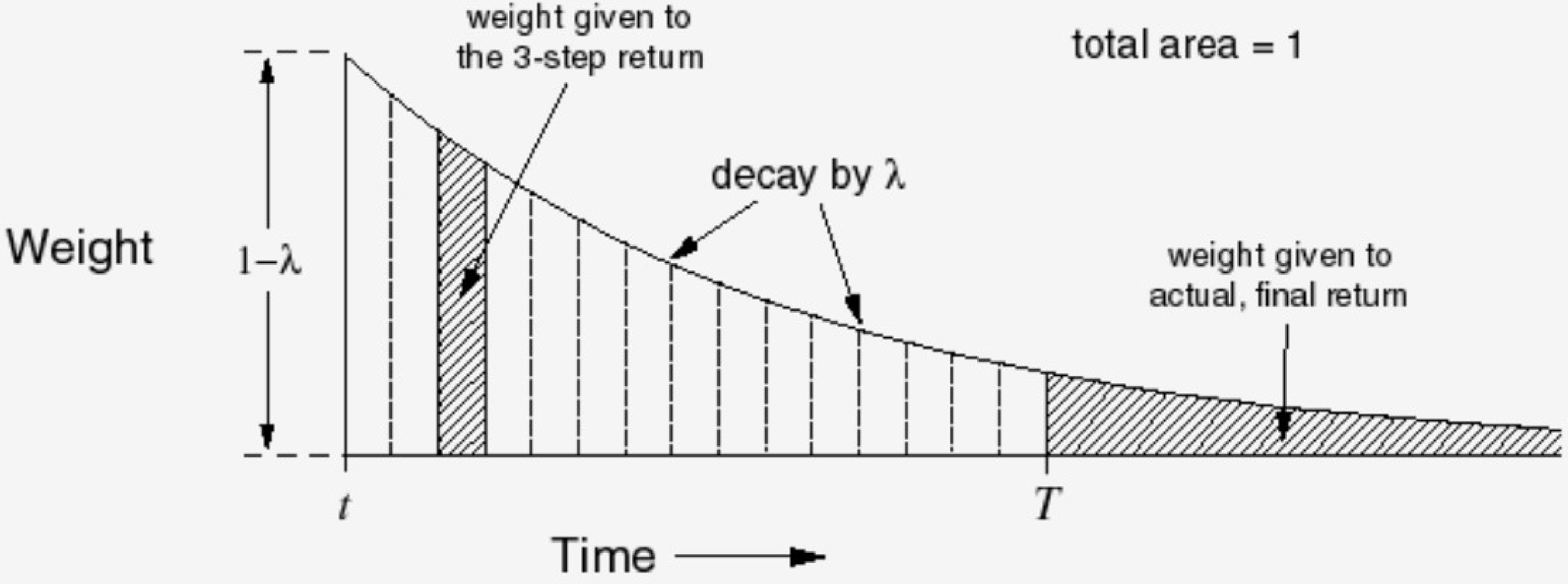
The λ-return

- Main idea: combine all n-step returns
- Definition: $$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$
- Update rule of TD(λ):

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^{\lambda} - V(S_t)\right)$$

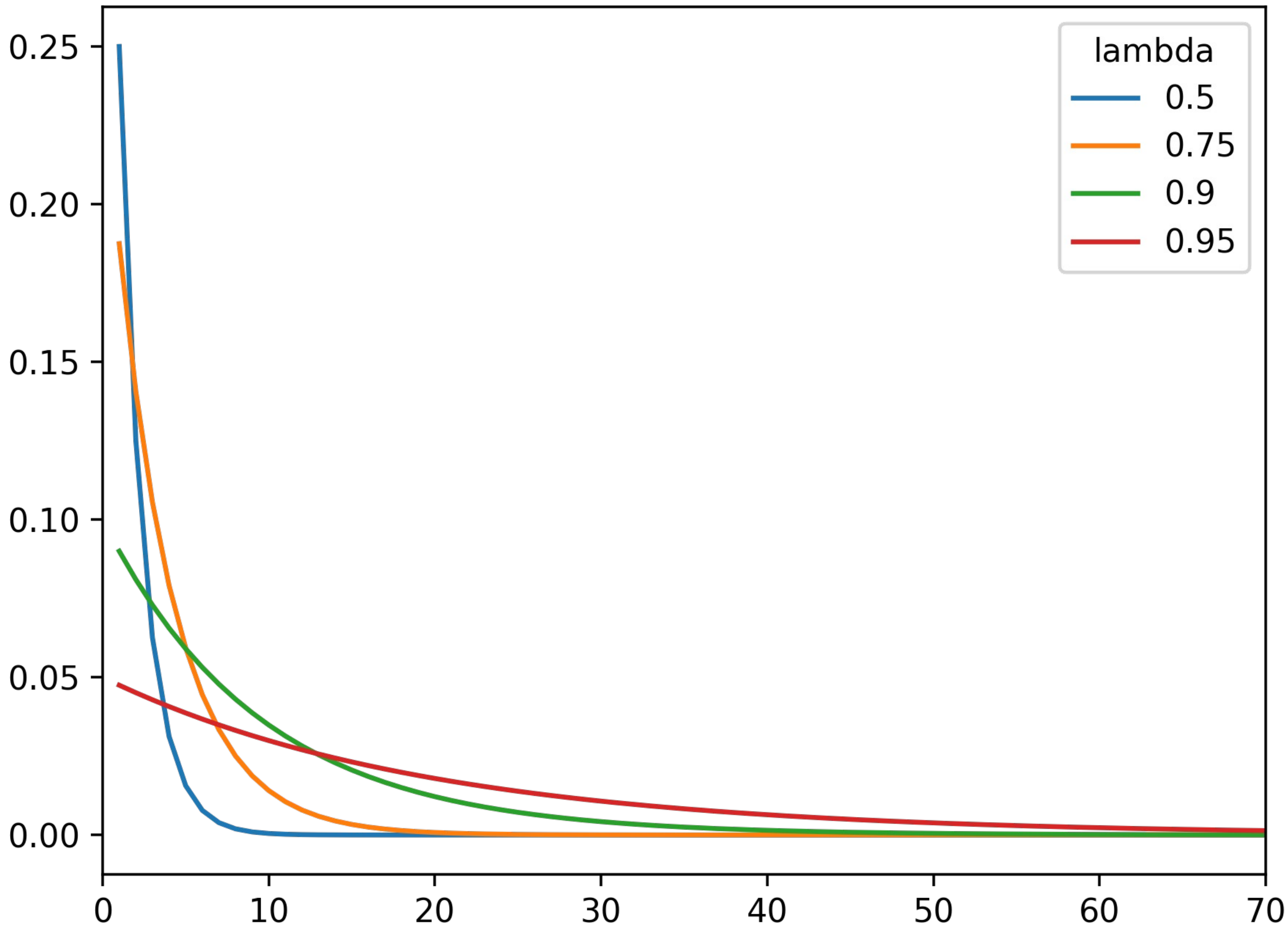


Weighting Function



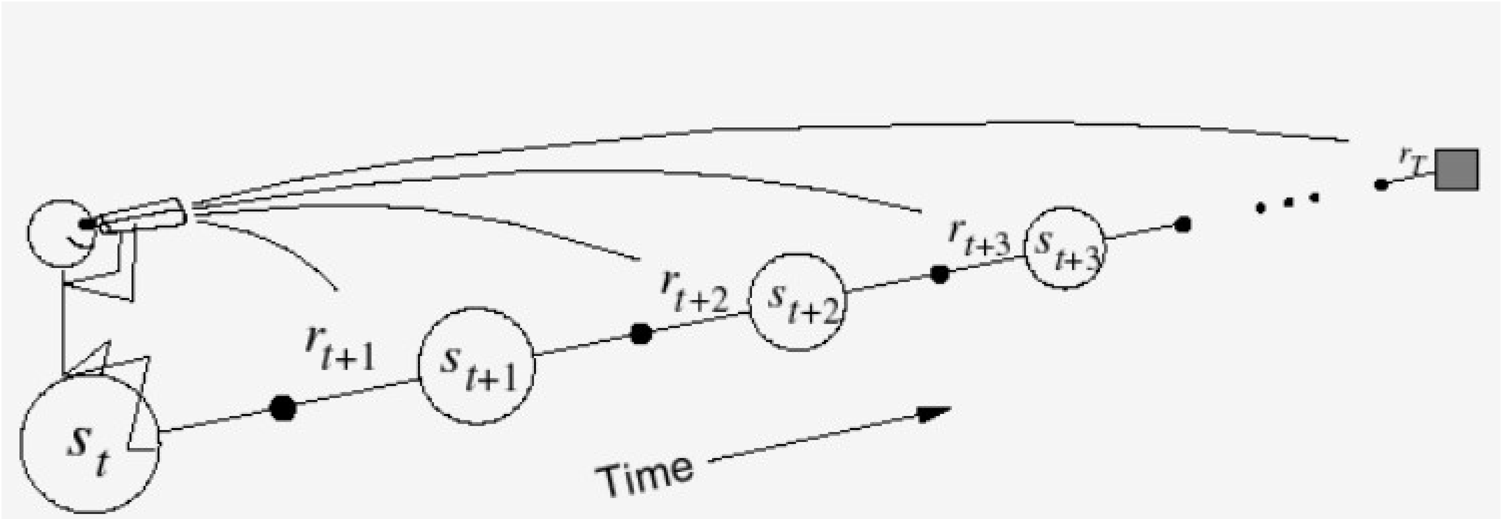
$$G_t^{\lambda} = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

Weighting Function



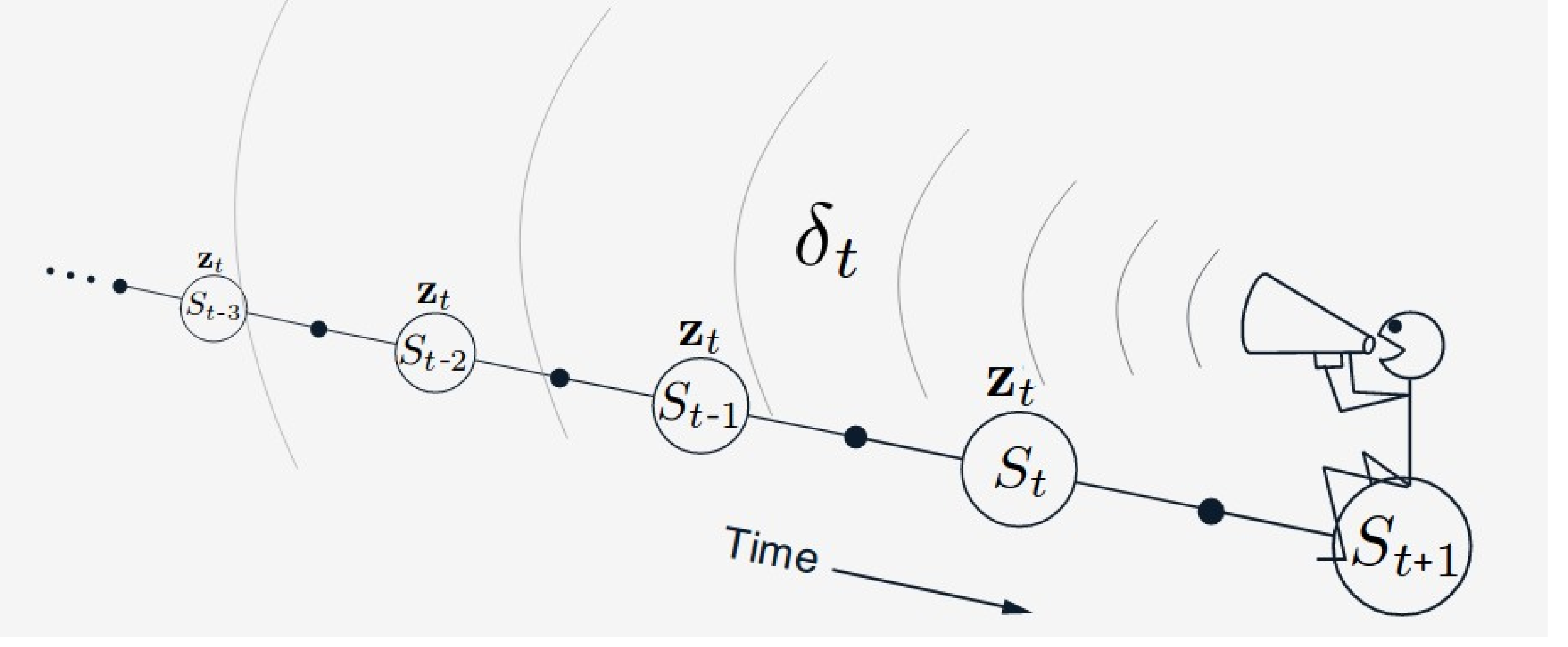
[https://amreis.github.io/ml/reinf-learn/2017/11/02/reinforcement-learning-eligibility-traces.html]

The “forward view”



- Updates value function towards the λ-return
- Looks into the future to compute the return
- Can only be computed from complete episodes

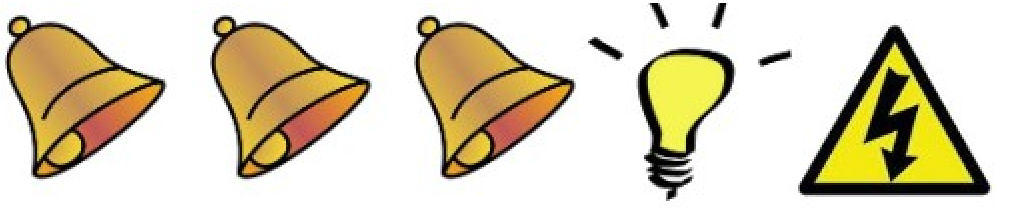
The “backward” view



- Forward view provides theory
- Backward view provides mechanism
- Update, online, after every step from incomplete episodes

The credit assignment problem

- Did the bell or the light cause the shock?



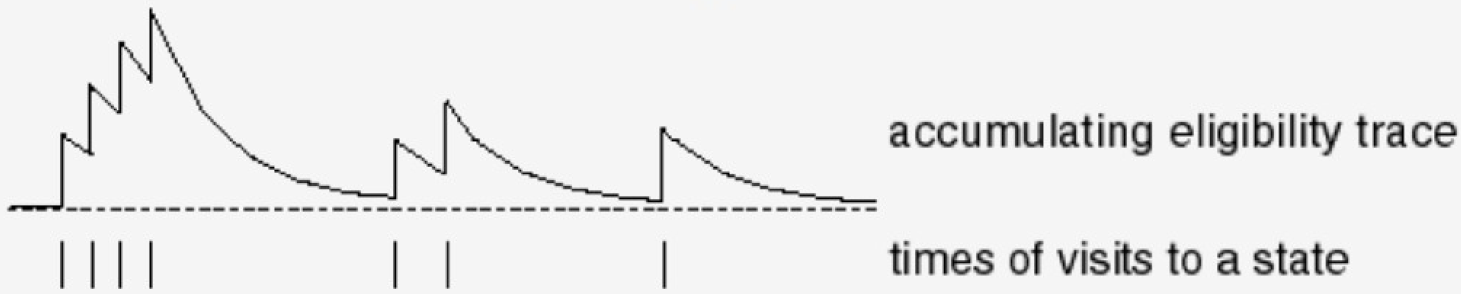
- Frequency heuristic: give credit to most frequent states
- Recency heuristic: give credit to most recent states

Eligibility Traces

- Eligibility traces combine both heuristics:

$$E_0(s) = 0$$

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$

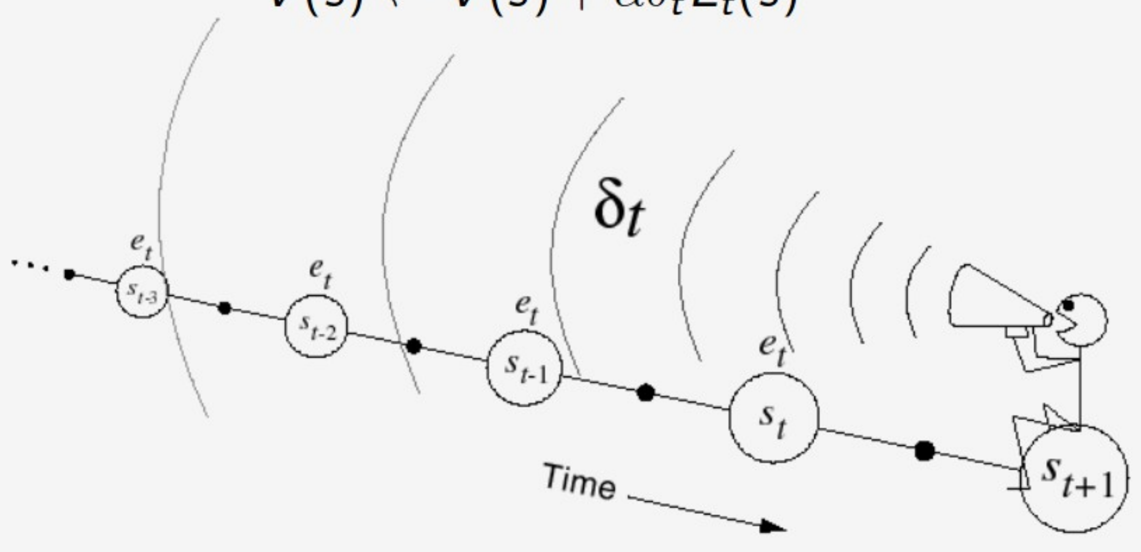


Backward view of TD(λ)

- Keep an eligibility trace for every state s
- Update value function for every state in proportion to TD-error and eligibility trace

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$



TD(λ) and TD(0)

- When λ = 0, only current state is updated:

$$E_t(s) = \mathbf{1}(S_t = s)$$
$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

- This is equivalent to the TD(0) update

TD(λ) and MC

- When λ = 1, credit is deferred until end of episode
- Works with episodic tasks with off-line updates

Online tabular TD(λ)

Initialize $V(s)$ arbitrarily and $e(s) = 0$, for all $s \in S$

Repeat (for each episode) :

Initialize s

Repeat (for each step of episode) :

$a \leftarrow$ action given by π for s

Take action a, observe reward, r, and next state s'

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

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

For all s :

$V(s) \leftarrow V(s) + \alpha\delta e(s)$

$e(s) \leftarrow \gamma\lambda e(s)$

$s \leftarrow s'$

Until s is terminal

[http://www-anw.cs.umass.edu/~barto/courses/cs687/Chapter%207.pdf]

Sarsa(λ)

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s,a

Repeat (for each episode) :

  Initialize s,a

  Repeat (for each step of episode) :

    Take action a, observe r,s'

    Choose a' from s' using policy derived from Q (e.g. ? - greedy)

    $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$

    $e(s,a) \leftarrow e(s,a) + 1$

    For all s,a:

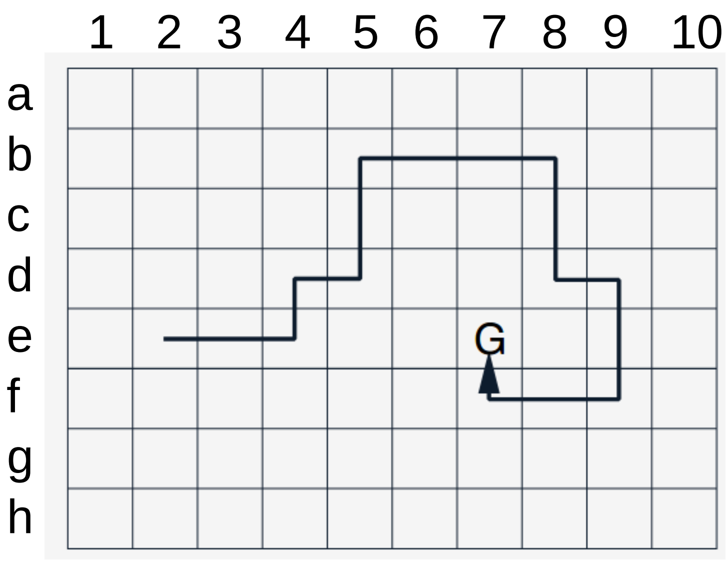
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$

      $e(s,a) \leftarrow \gamma\lambda e(s,a)$

    $s \leftarrow s'; a \leftarrow a'$

  Until s is terminal

Walk-through



Actions: L,R,U,D

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s,a

Repeat (for each episode) :

  Initialize s, a

  Repeat (for each step of episode) :

    Take action a, observe r, s'

    Choose a' from s' using policy derived from Q (e.g. ? - greedy)

    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$

    $e(s,a) \leftarrow e(s,a) + 1$

    For all s,a :

      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$

      $e(s, a) \leftarrow \gamma \lambda e(s, a)$

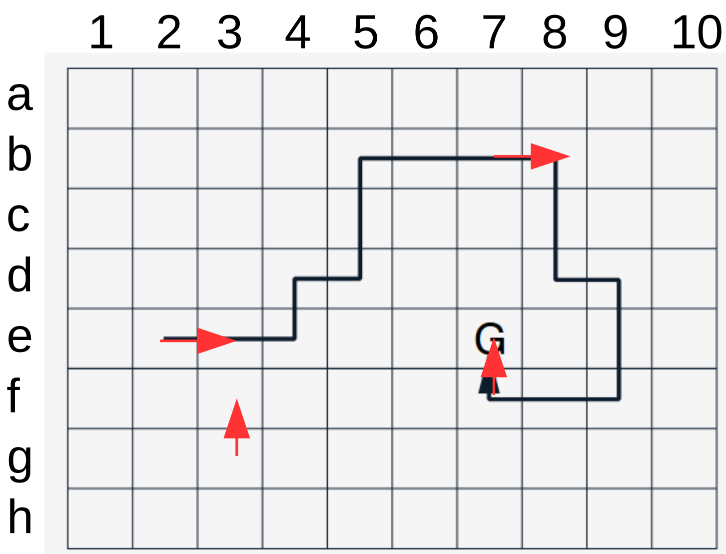
    $s \leftarrow s'; a \leftarrow a'$

  Until s is terminal

In small groups, compute the updates to Q(<f,7>,U), Q(<b,7>,R), Q(<e,2>,R) and Q(<g,3>,U) assuming:

Discount factor γ = 0.95
Goal reward = 100
λ = 0.95

Walk-through



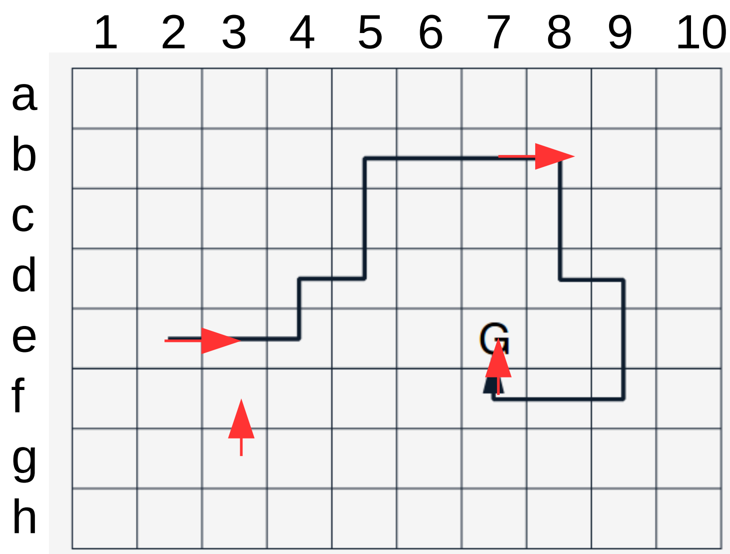
Actions: L,R,U,D

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s,a
Repeat (for each episode) :
  Initialize s, a
  Repeat (for each step of episode) :
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g. ? - greedy)
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all s,a :
      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$
      $e(s, a) \leftarrow \gamma \lambda e(s, a)$
    $s \leftarrow s'; a \leftarrow a'$
  Until s is terminal

In small groups, compute the updates to Q(<f,7>,U), Q(<b,7>,R), Q(<e,2>,R) and Q(<g,3>,U) assuming:

Discount factor γ = 0.95
Goal reward = 100
λ = 0.95

What did you get?



Actions: L,R,U,D

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s,a

Repeat (for each episode) :

  Initialize s,a

  Repeat (for each step of episode) :

    Take action a, observe r, s'

    Choose a' from s' using policy derived from Q (e.g. ? - greedy)

    $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$

    $e(s,a) \leftarrow e(s,a) + 1$

    For all s,a :

      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$

      $e(s,a) \leftarrow \gamma \lambda e(s,a)$

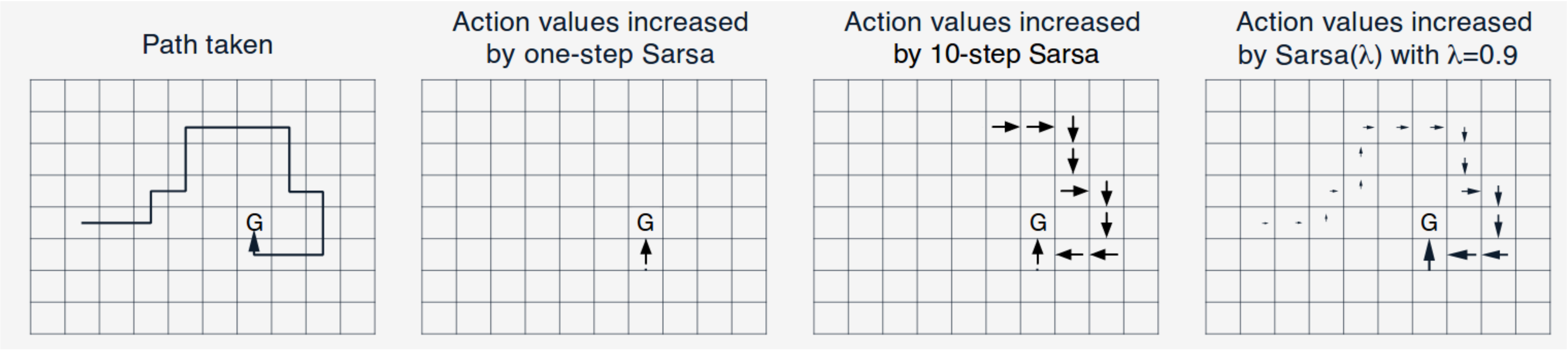
    $s \leftarrow s'; a \leftarrow a'$

  Until s is terminal

In small groups, compute the updates to Q(<f,7>,U), Q(<b,7>,R), Q(<e,2>,R) and Q(<g,3>,U) assuming:

Discount factor γ = 0.95
Goal reward = 100
λ = 0.95

Comparison



Path taken
G
Action values increased by one-step Sarsa
G
Action values increased by 10-step Sarsa
G
Action values increased by Sarsa(λ) with λ=0.9
G

Project Planning Breakout

- Meet with partner(s) if working in group
- Plan out the activities for this week – make concrete goals that you want to accomplish
- Find research articles relevant to your project
- Write down any questions for me