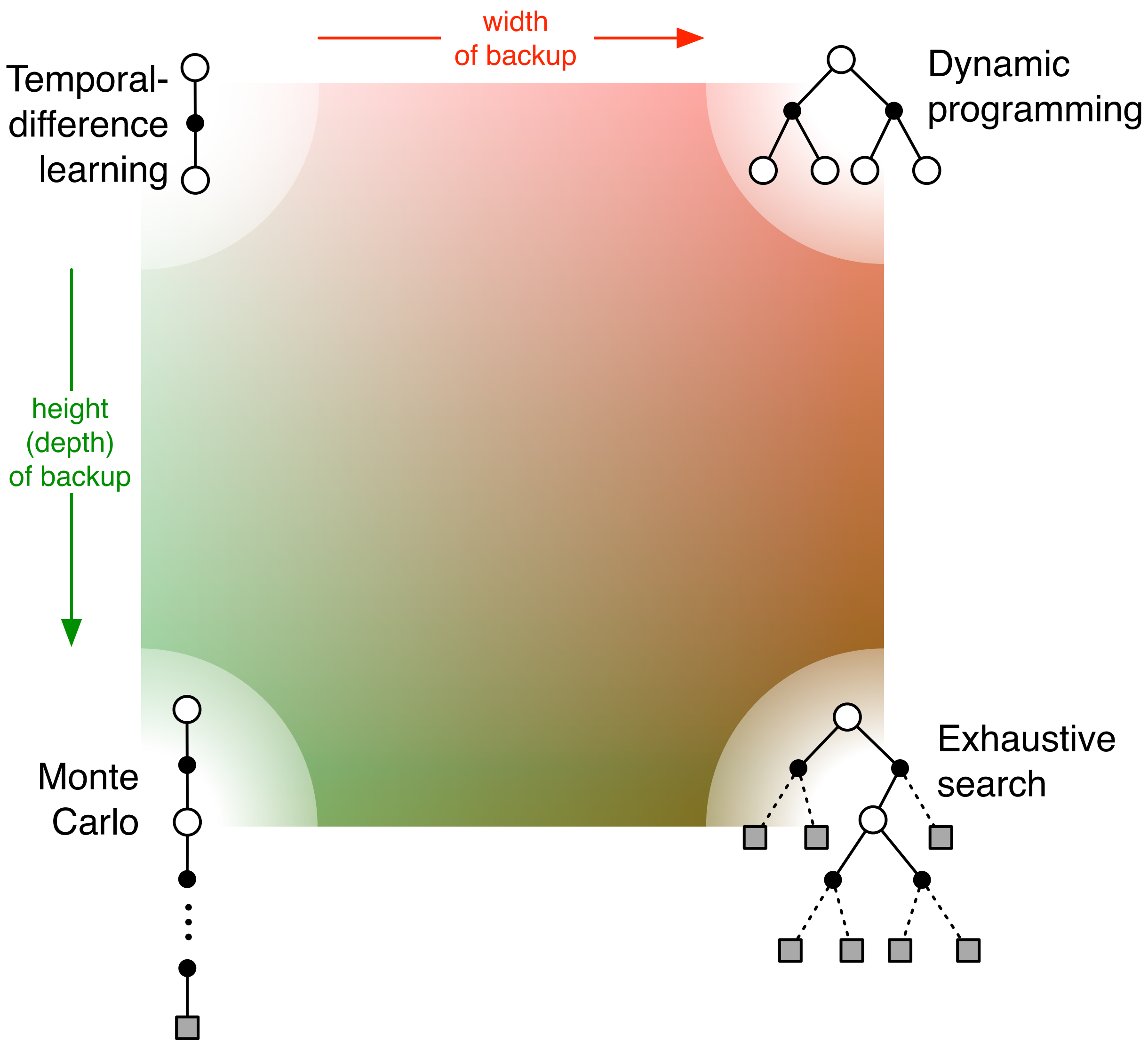
Eligibility Traces

Unifying Monte Carlo and TD

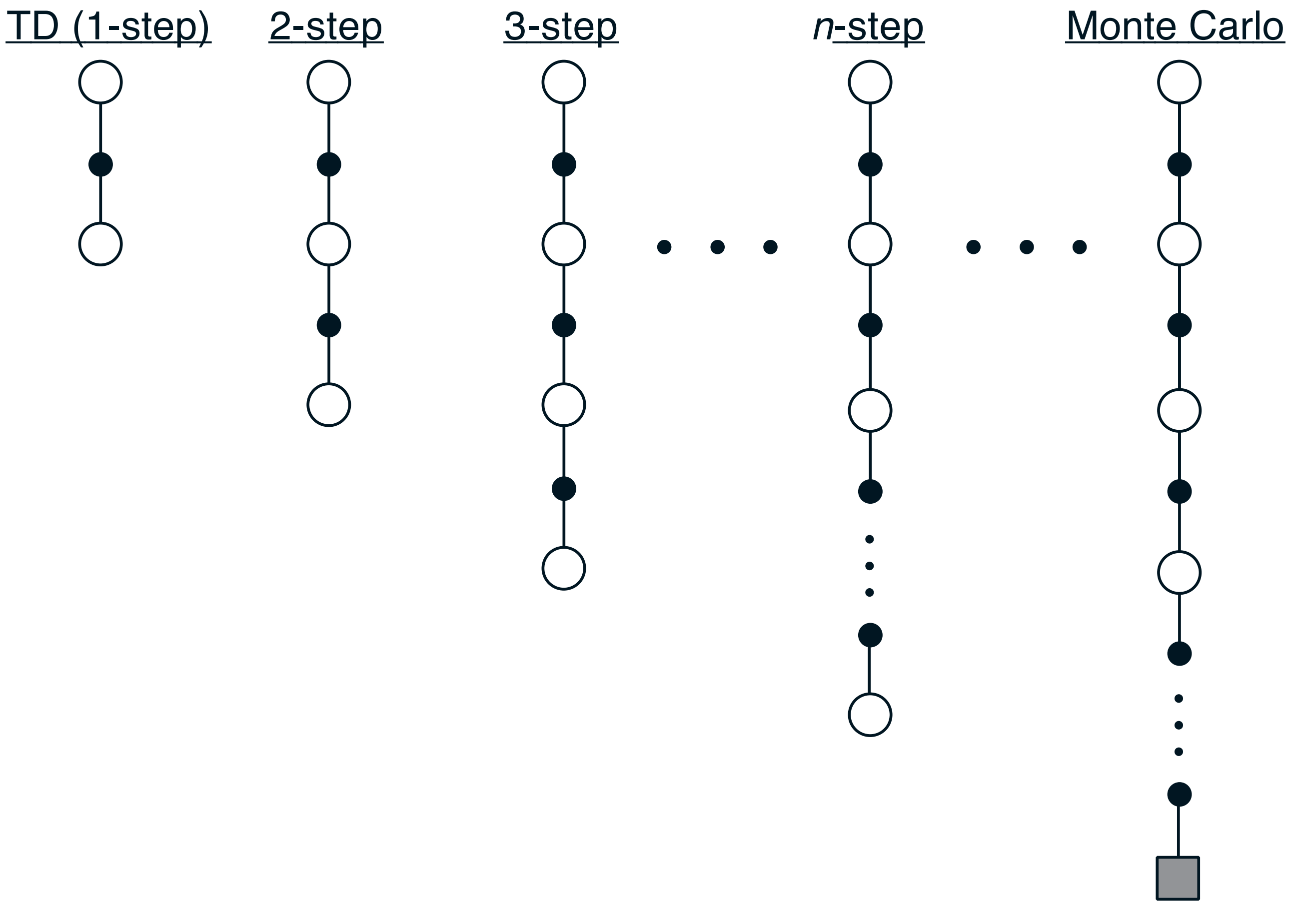
key algorithms: TD(λ), Sarsa(λ), Q(λ)

Unified View



N-step TD Prediction

- Idea: Look farther into the future when you do TD backup (1, 2, 3, …, n steps)



Mathematics of N-step TD Prediction

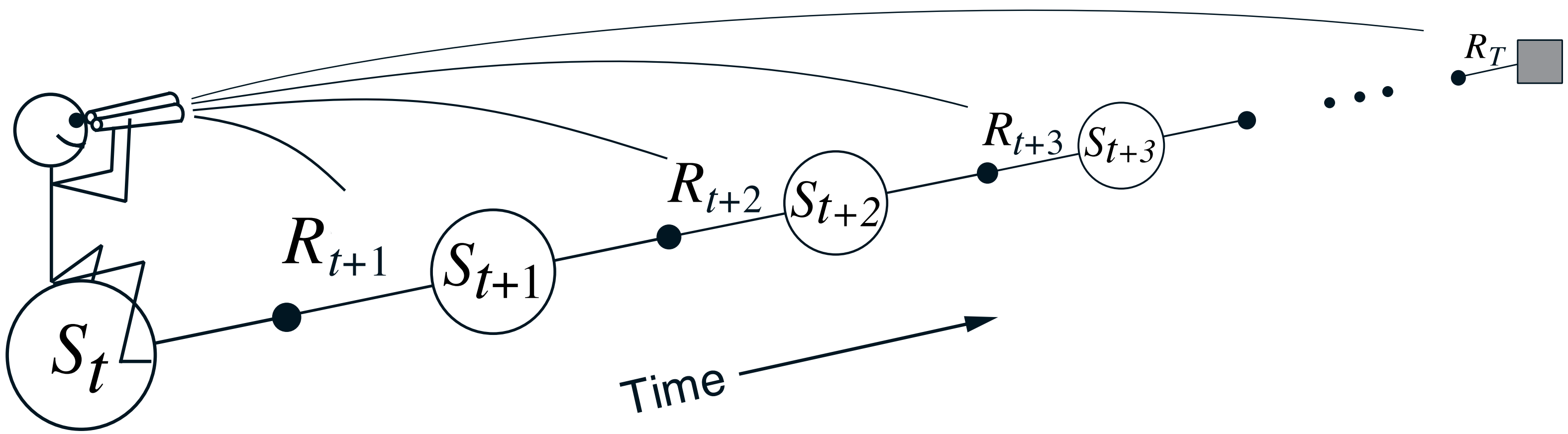
- Monte Carlo: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$

- TD: $G_t^{(1)} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
  - Use V_t to estimate remaining return

- *n*-step TD:
  - 2 step return: $G_t^{(2)} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_t(S_{t+2})$
  - *n*-step return: $G_t^{(n)} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_t(S_{t+n})$

Forward View of TD(λ)

- Look forward from each state to determine update from future states and rewards:



Learning with *n*-step Backups

- Backup computes an increment:

$$\Delta_t(S_t) \doteq \alpha\Big[G_t^{(n)} - V_t(S_t)\Big] \qquad \Delta_t(s) = 0, \forall s \neq S_t$$

- Then,
  - Online updating:

$$V_{t+1}(s) = V_t(s) + \Delta_t(s), \qquad \forall s \in \mathcal{S}$$

  - Off-line updating:

$$V(s) \leftarrow V(s) + \sum_{t=0}^{T-1} \Delta_t(s) \qquad \forall s \in \mathcal{S}$$

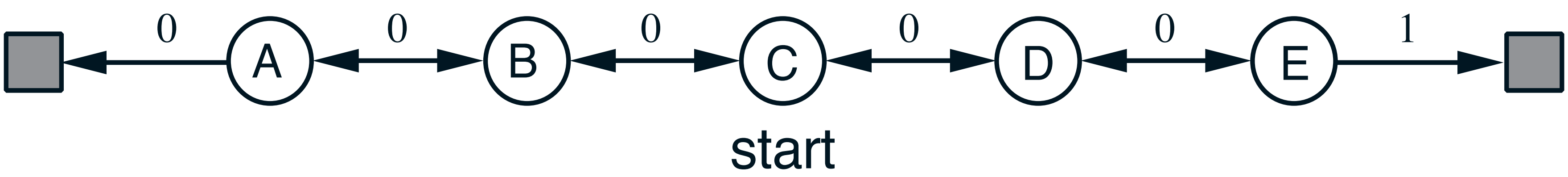
Error-reduction property

- Error reduction property of n-step returns

$$\underbrace{\max_s \left| \mathbb{E}_\pi\left[G_t^{(n)} \middle| S_t = s \right] - v_\pi(s) \right|}_{\text{Maximum error using } n\text{-step return}} \leq \gamma^n \underbrace{\max_s \left| V_t(s) - v_\pi(s) \right|}_{\text{Maximum error using V}}$$

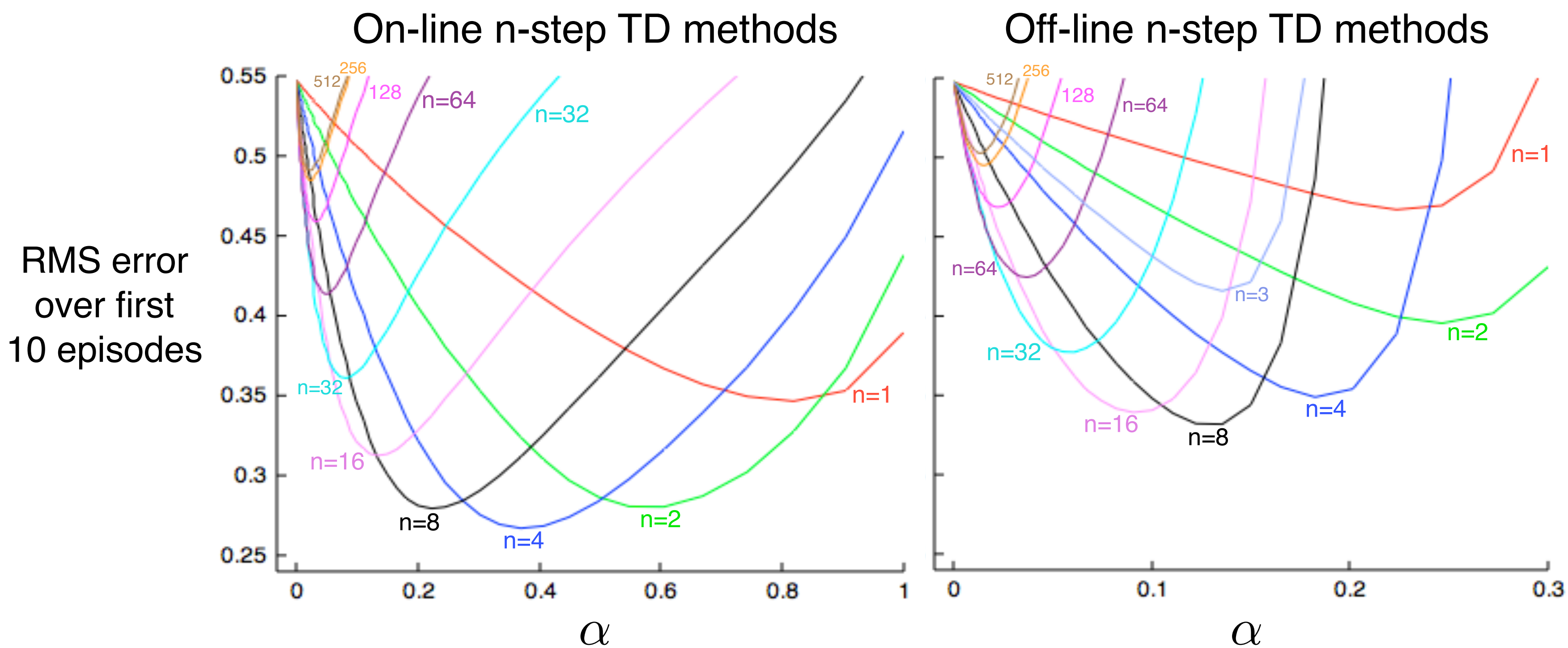
- Using this, you can show that n-step methods converge

Random Walk Examples



- How does 2-step TD work here?
- How about 3-step TD?

A Larger Example – 19-state Random Walk



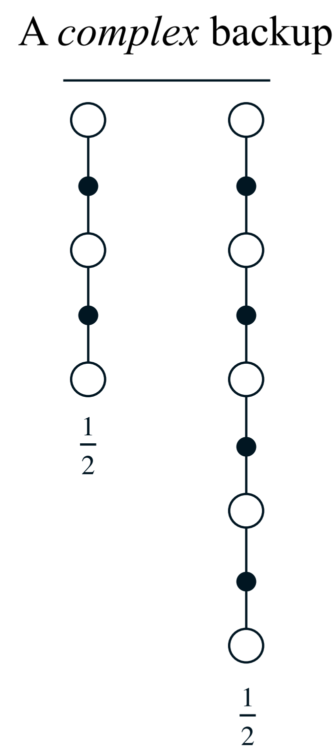
- On-line is better than off-line
- An *intermediate n* is best
- Do you think there is an optimal *n*? for every task?

Averaging N-step Returns

- n-step methods were introduced to help with TD(λ) understanding
- Idea: backup an average of several returns
  - e.g. backup half of 2-step and half of 4-step

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

- Called a complex backup
  - Draw each component
  - Label with the weights for that component



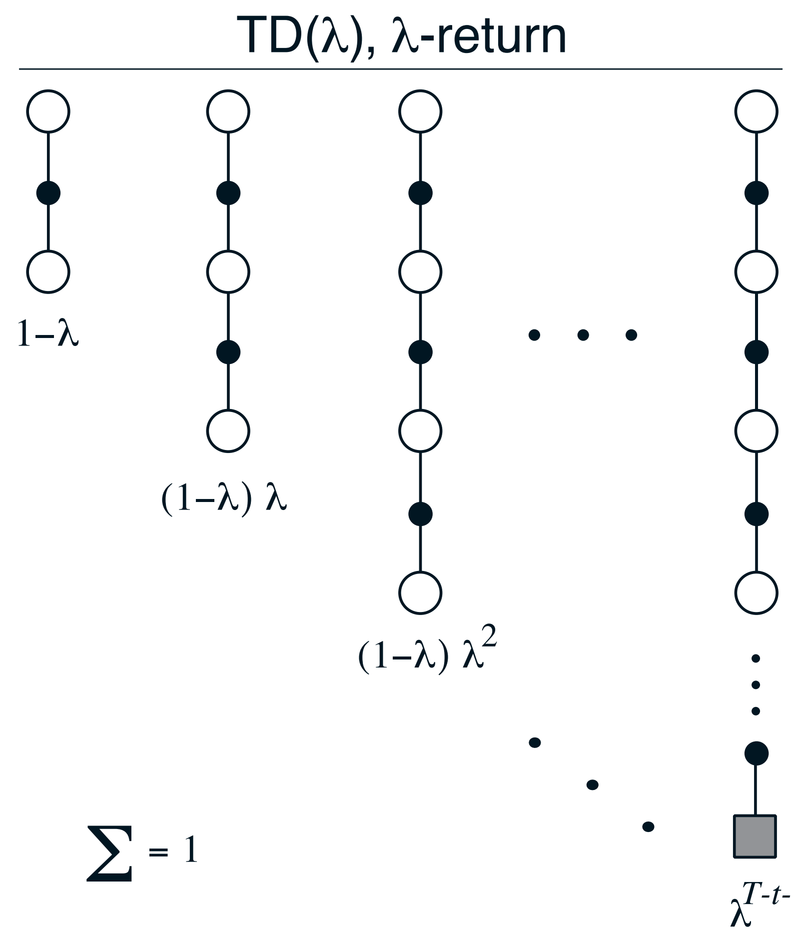
Forward View of TD(λ)

- TD(λ) is a method for averaging all *n*-step backups
  - weight by λ^{n-1} (time since visitation)
  - λ-return:

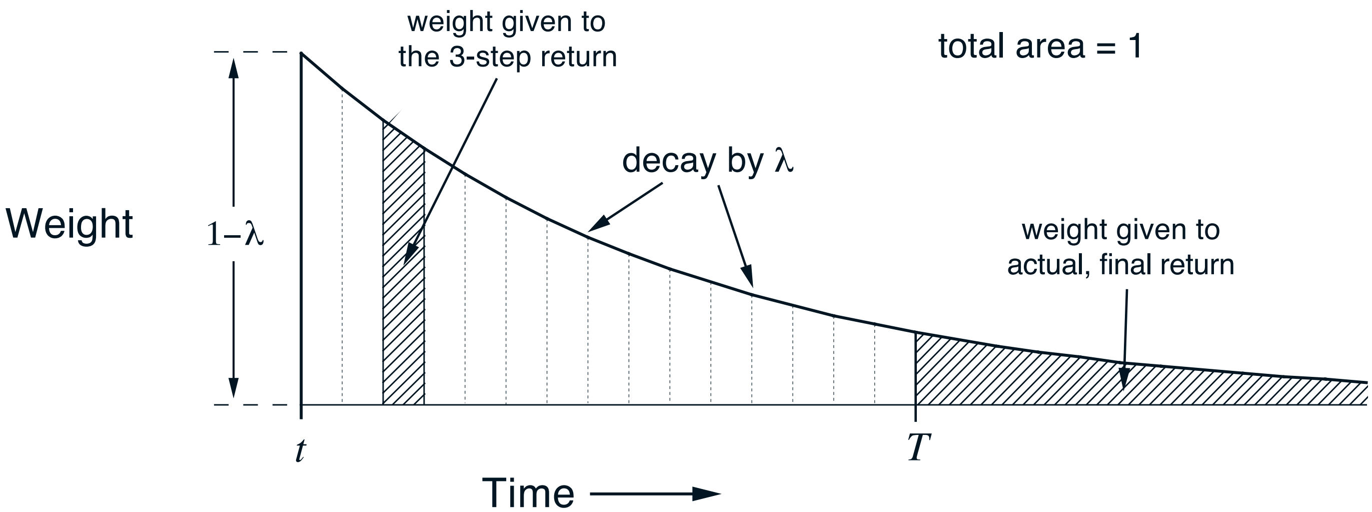
$$G_t^{\lambda} \doteq (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

- Backup using λ-return:

$$\Delta_t(S_t) \doteq \alpha\Big[G_t^{\lambda} - V_t(S_t)\Big]$$



λ-return Weighting Function



$$G_t^{\lambda} = \underbrace{(1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{(n)}}_{\text{Until termination}} + \underbrace{\lambda^{T-t-1} G_t}_{\text{After termination}}$$

Relation to TD(0) and MC

- The λ-return can be rewritten as:

$$G_t^\lambda = \underbrace{(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_t^{(n)}}_{\text{Until termination}} + \underbrace{\lambda^{T-t-1}G_t}_{\text{After termination}}$$

- If λ = 1, you get MC:

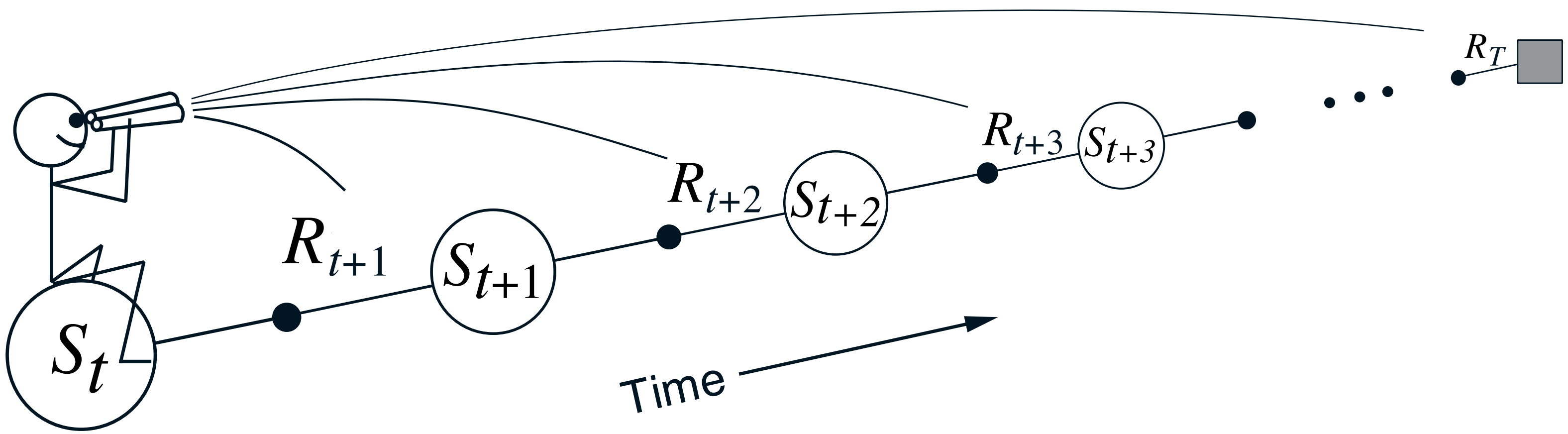
$$G_t^\lambda = (1-1)\sum_{n=1}^{T-t-1}1^{n-1}G_t^{(n)} + 1^{T-t-1}G_t = G_t$$

- If λ = 0, you get TD(0)

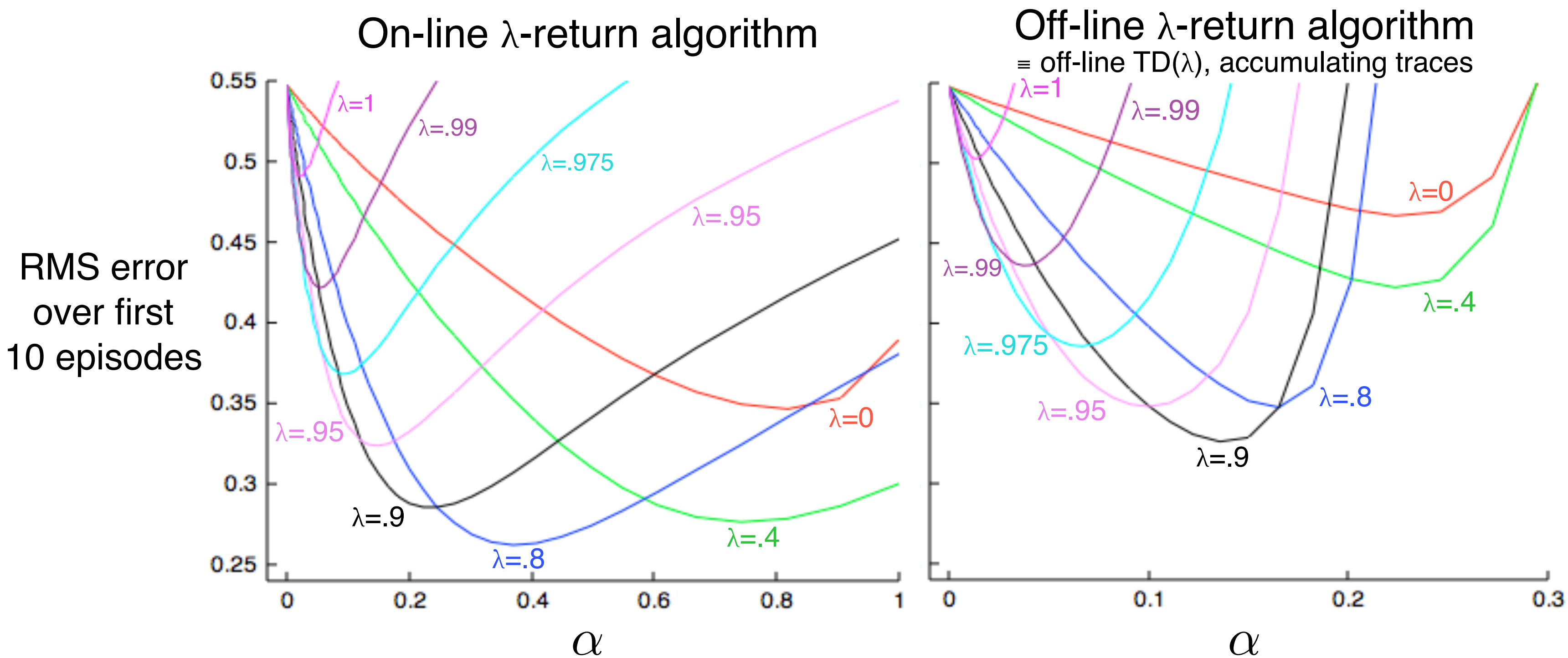
$$G_t^\lambda = (1-0)\sum_{n=1}^{T-t-1}0^{n-1}G_t^{(n)} + 0^{T-t-1}G_t = G_t^{(1)}$$

Forward View of TD(λ)

- Look forward from each state to determine update from future states and rewards:



λ-return on the Random Walk

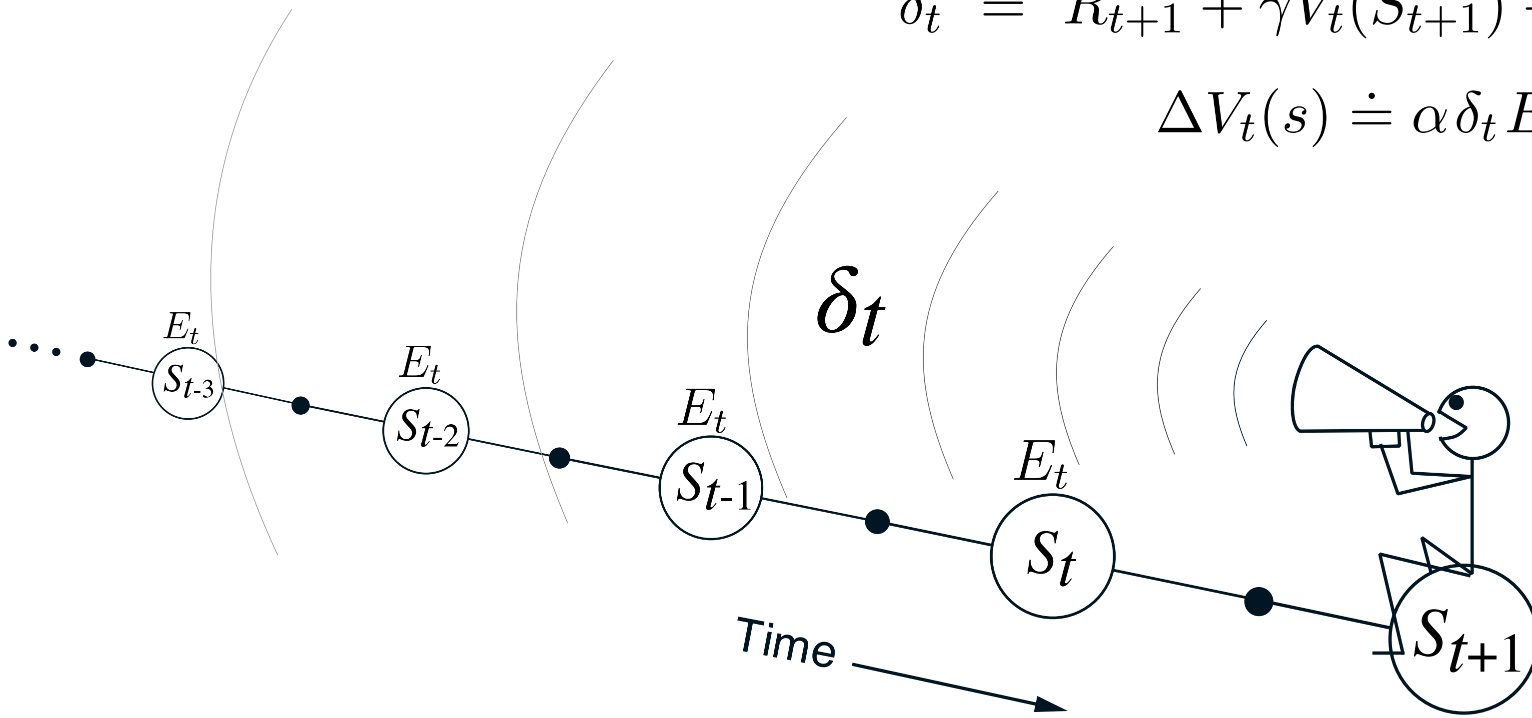


- On-line >> Off-line
- Intermediate values of λ best
- λ-return better than *n*-step return

Backward View

$$\delta_t \doteq R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)$$

$$\Delta V_t(s) \doteq \alpha \delta_t E_t(s)$$



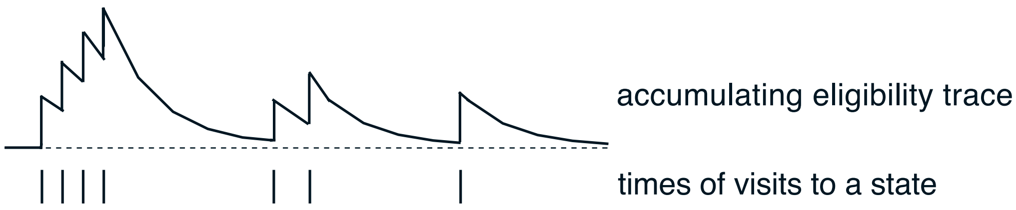
- Shout δ_t backwards over time
- The strength of your voice decreases with temporal distance by $\gamma\lambda$

Backward View of TD(λ)

- The forward view was for theory
- The backward view is for *mechanism*

- New variable called *eligibility trace* $E_t(s) \in \mathbb{R}^+$
  - On each step, decay all traces by γλ and increment the trace for the current state by 1
  - *Accumulating trace*

$$E_t(s) = \begin{cases} \gamma\lambda E_{t-1}(s) & \text{if } s \neq S_t; \\ \gamma\lambda E_{t-1}(s) + 1 & \text{if } s = S_t, \end{cases}$$



On-line Tabular TD(λ)

Initialize $V(s)$ arbitrarily (but set to 0 if s is terminal)
Repeat (for each episode):
    Initialize $E(s) = 0$, for all $s \in \mathcal{S}$
    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'
        $\delta \leftarrow R + \gamma V(S') - V(S)$
        $E(S) \leftarrow E(S) + 1$ (accumulating traces)
        or $E(S) \leftarrow (1 - \alpha)E(S) + 1$ (dutch traces)
        or $E(S) \leftarrow 1$ (replacing traces)
        For all $s \in \mathcal{S}$:
            $V(s) \leftarrow V(s) + \alpha \delta E(s)$
            $E(s) \leftarrow \gamma \lambda E(s)$
        $S \leftarrow S'$
    until S is terminal

Relation of Backwards View to MC & TD(0)

- Using update rule:

$$\Delta V_t(s) \doteq \alpha \delta_t E_t(s)$$

- As before, if you set λ to 0, you get to TD(0)
- If you set λ to 1, you get MC but in a better way
  - Can apply TD(1) to continuing tasks
  - Works incrementally and on-line (instead of waiting to the end of the episode)

Forward View = Backward View

- The forward (theoretical) view of TD(λ) is equivalent to the backward (mechanistic) view for off-line updating

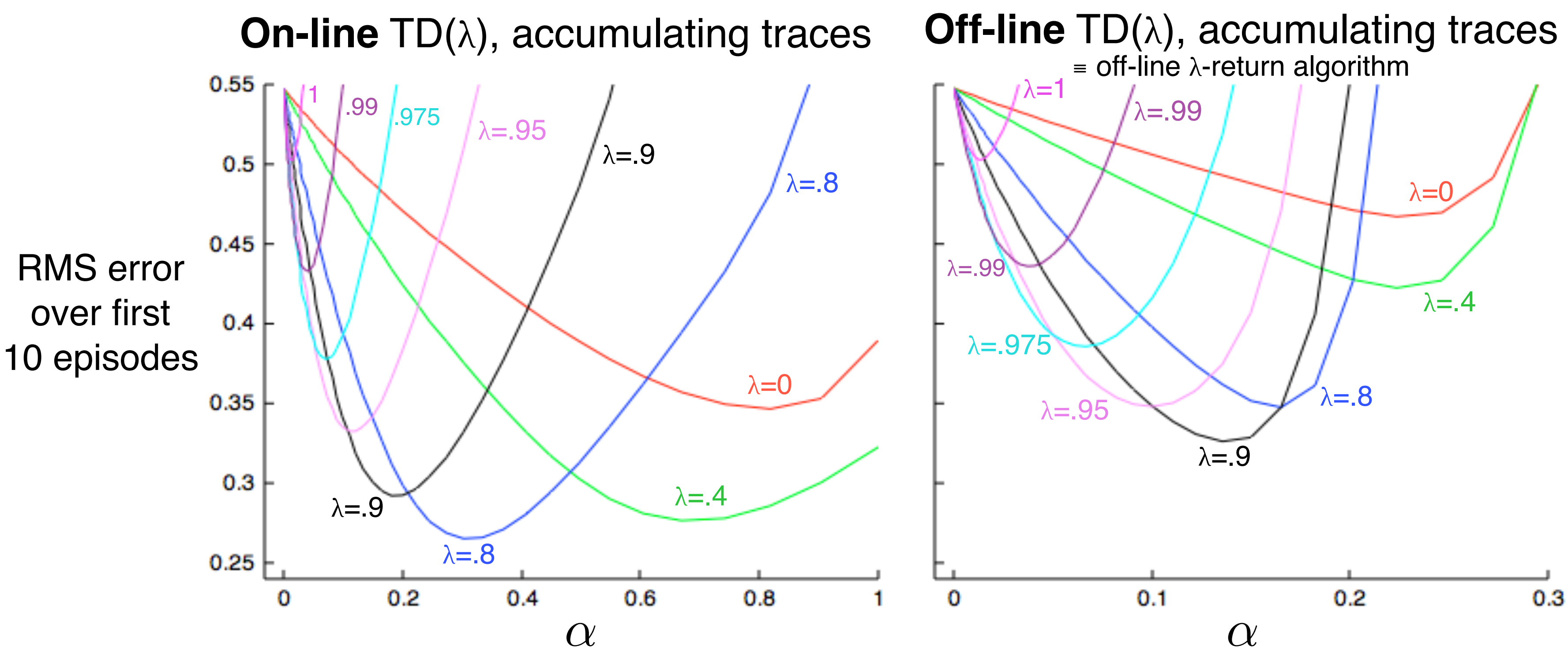
$$\underbrace{\sum_{t=0}^{T-1} \Delta V_t^{TD}(s)}_{\text{Backward updates}} = \underbrace{\sum_{t=0}^{T-1} \Delta V_t^{\lambda}(S_t) I_{sS_t}}_{\text{Forward updates}}$$

⇓ algebra

$$\sum_{t=0}^{T-1} \alpha \, I_{sS_t} \sum_{k=t}^{T-1} (\gamma\lambda)^{k-t} \delta_k$$

- On-line updating with small α is similar

On-line versus Off-line on Random Walk



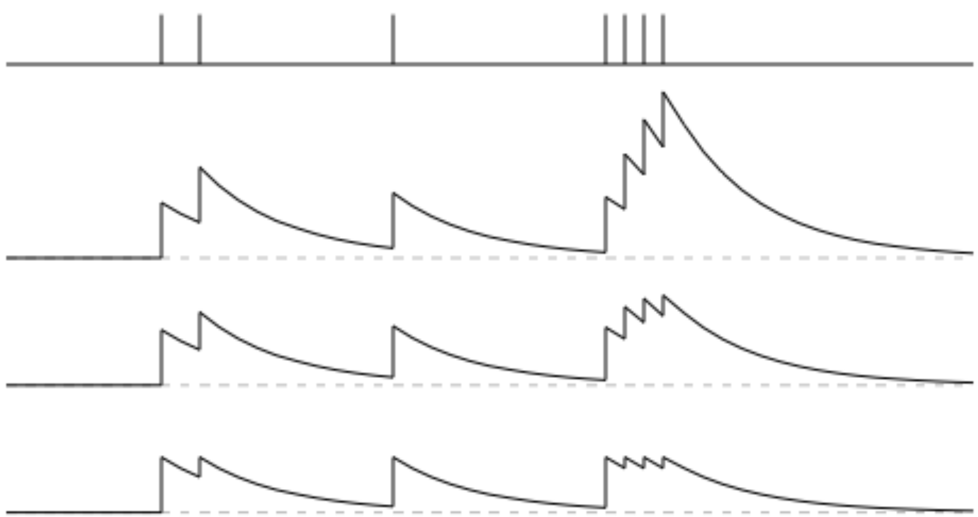
- Same 19 state random walk
- On-line performs better over a broader range of parameters

Replacing and Dutch Traces

- All traces fade the same:

$$E_t(s) \doteq \gamma\lambda E_{t-1}(s), \qquad \forall s \in \mathcal{S}, s \neq S_t$$

- But increment differently!



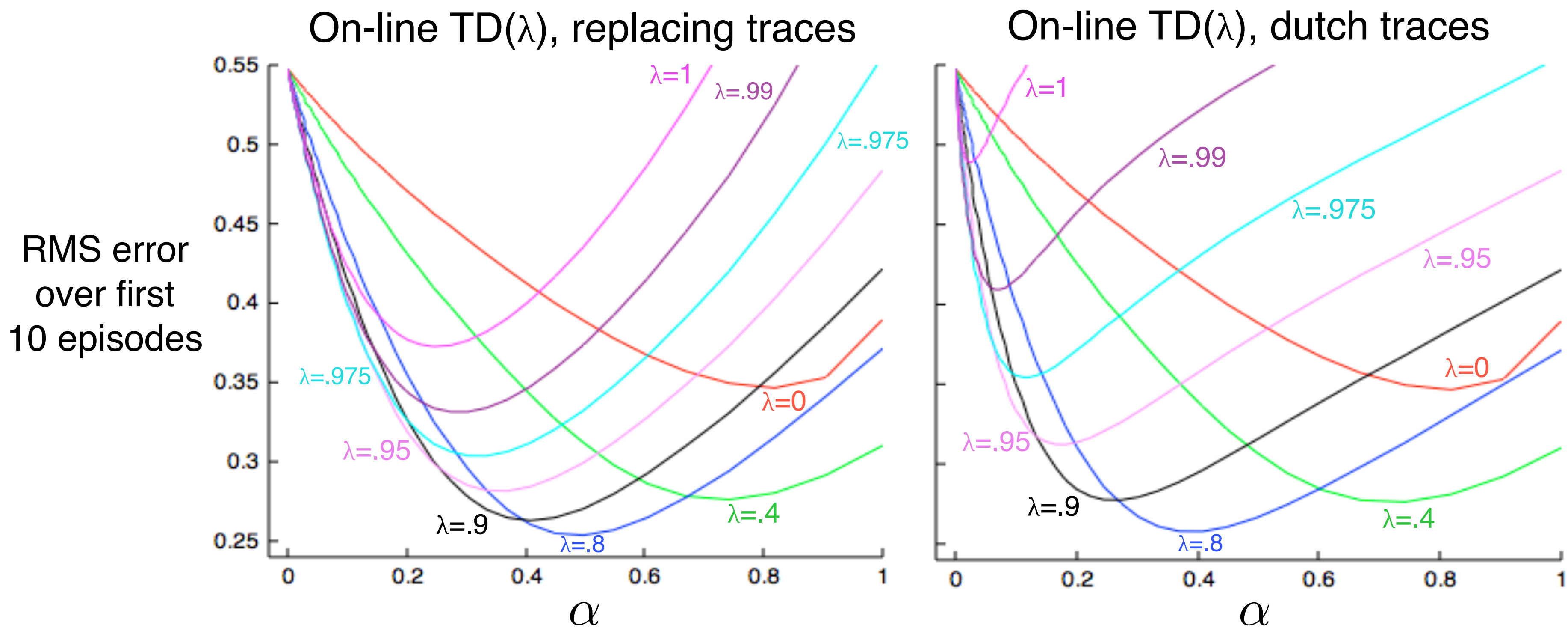
times of state visits

accumulating traces $E_t(S_t) \doteq \gamma\lambda E_{t-1}(S_t) + 1$

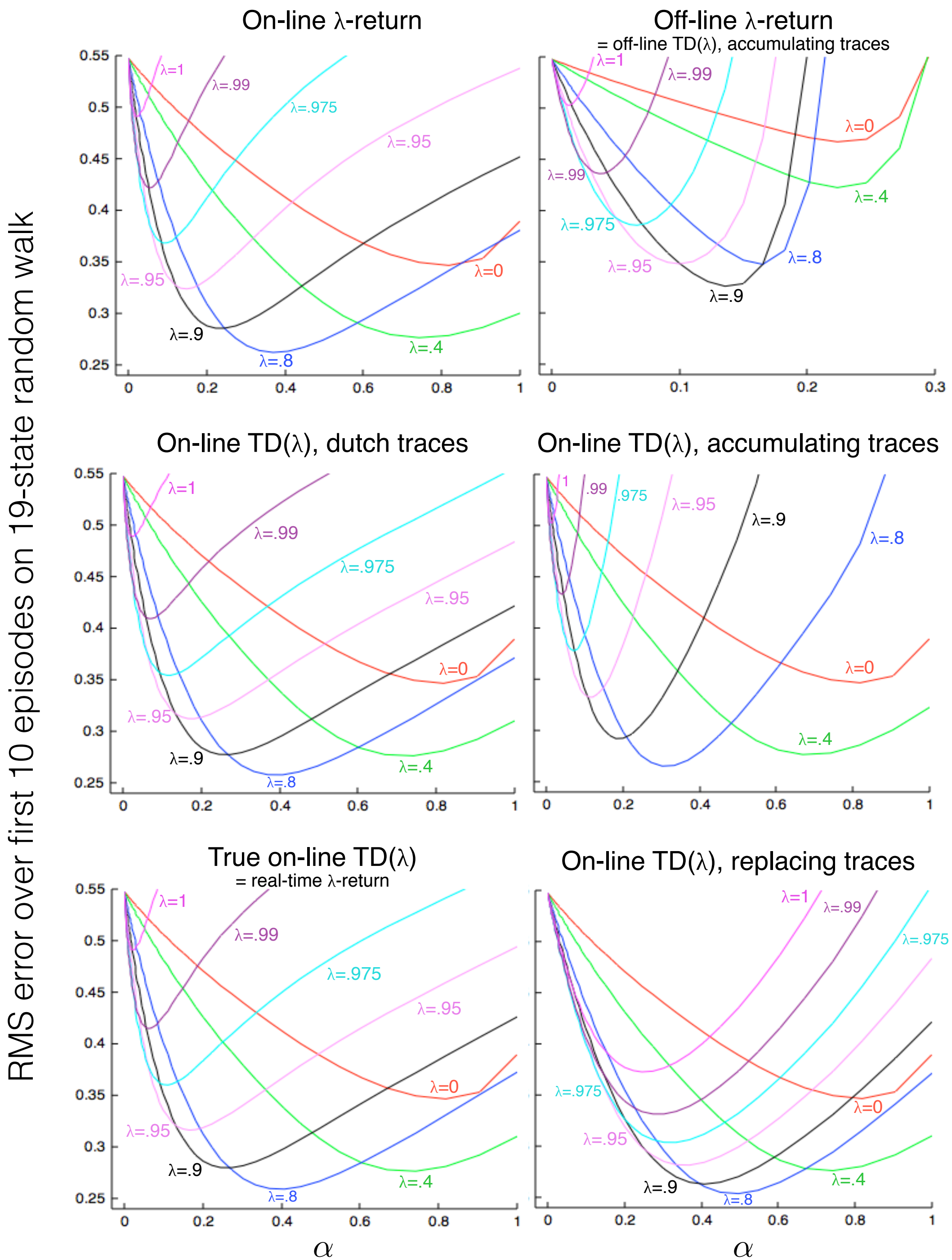
dutch traces $E_t(S_t) \doteq (1-\alpha)\gamma\lambda E_{t-1}(S_t) + 1$

replacing traces $E_t(S_t) \doteq 1$

Replacing and Dutch on the Random Walk

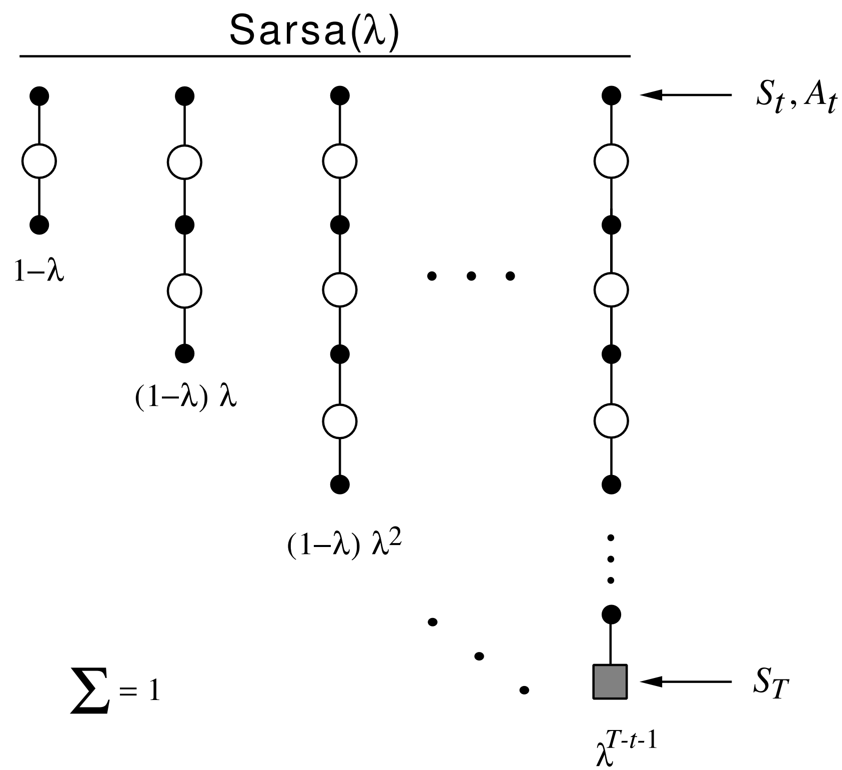


All λ results on the random walk



Control: Sarsa(λ)

- Everything changes from states to state-action pairs



$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \delta_t E_t(s,a), \quad \forall s, a$$

where

$$\delta_t = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t)$$

and

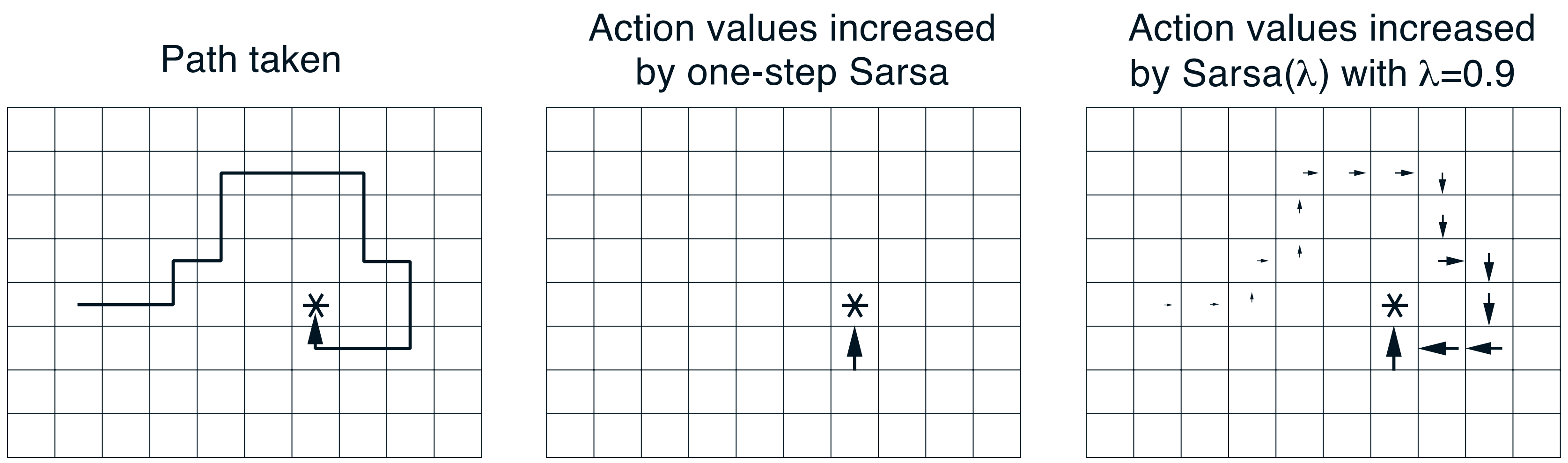
$$E_t(s,a) = \begin{cases} \gamma\lambda E_{t-1}(s,a) + 1 & \text{if } s = S_t \text{ and } a = A_t; \\ \gamma\lambda E_{t-1}(s,a) & \text{otherwise.} \end{cases} \quad \text{for all } s, a$$

Demo

Sarsa(λ) Algorithm

Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
    $E(s, a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    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., ε-greedy)
        $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
        $E(S, A) \leftarrow E(S, A) + 1$
        For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
            $Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$
            $E(s, a) \leftarrow \gamma \lambda E(s, a)$
        $S \leftarrow S'$; $A \leftarrow A'$
    until S is terminal

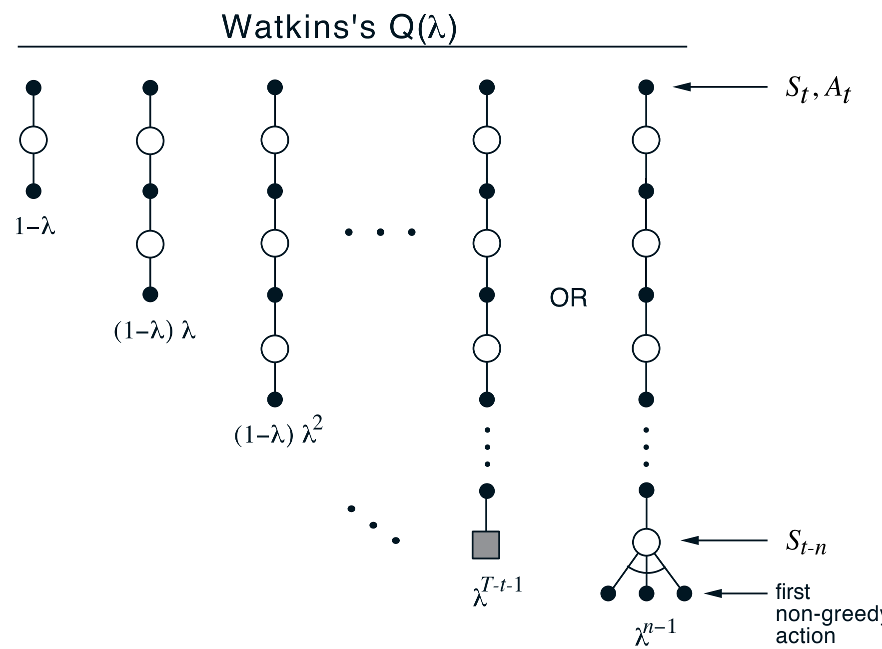
Sarsa(λ) Gridworld Example



- With one trial, the agent has much more information about how to get to the goal
  - not necessarily the *best* way
- Can considerably accelerate learning

Watkins's Q(λ)

- How can we extend this to Q-learning?
- If you mark every state action pair as eligible, you backup over non-greedy policy
- *Watkins's*: Zero out eligibility trace after a non-greedy action. Do max when backing up at first non-greedy choice.



$$Z_t(s,a) = \begin{cases} 1 + \gamma\lambda E_{t-1}(s,a) & \text{if } S_t = s,\ A_t = a, \text{ and } A_t \text{ was greedy;} \\ 0 & \text{if } S_t = s,\ A_t = a, \text{ and } A_t \text{ was not greedy;} \\ \gamma\lambda E_{t-1}(s,a) & \text{for all other } s, a; \end{cases}$$

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t E_t(s,a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

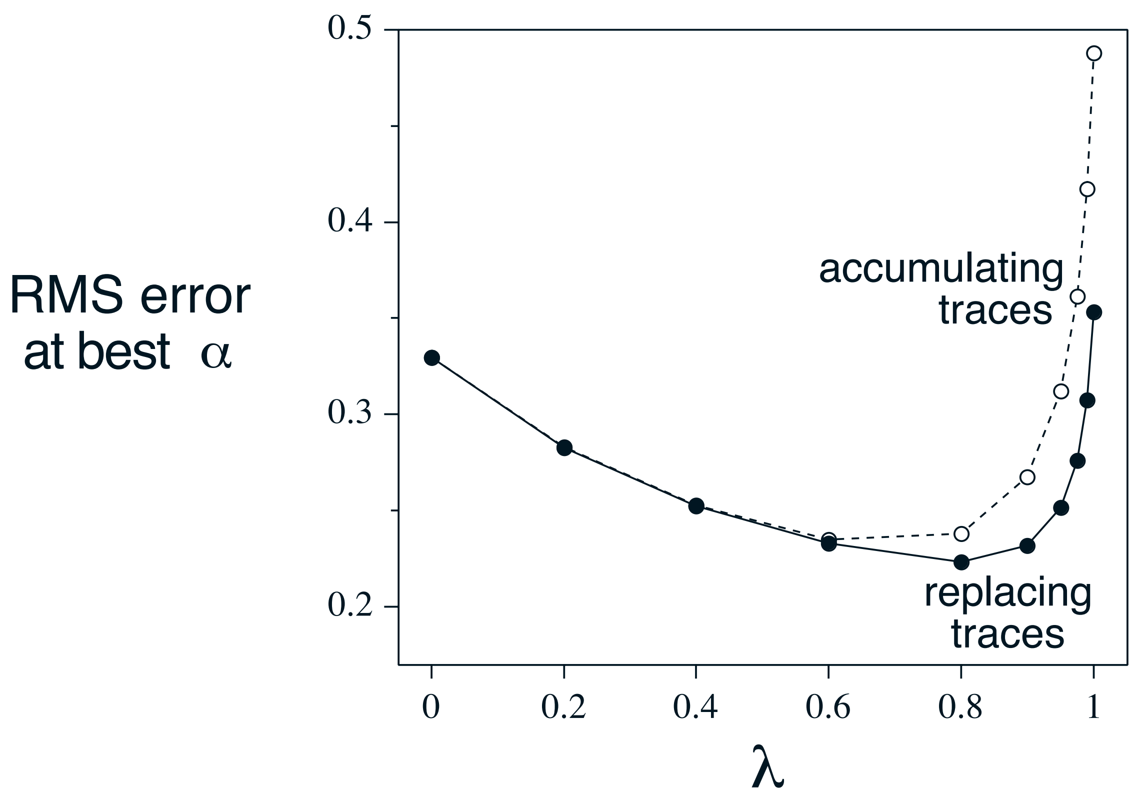
$$\delta_t = R_{t+1} + \gamma \max_{a'} Q_t(S_{t+1}, a') - Q_t(S_t, A_t)$$

Watkins's Q(λ)

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
    $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    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., ε-greedy)
        $A^* \leftarrow \arg\max_a Q(S',a)$ (if A' ties for the max, then $A^* \leftarrow A'$)
        $\delta \leftarrow R + \gamma Q(S', A^*) - Q(S,A)$
        $E(S,A) \leftarrow E(S,A) + 1$
        For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
            $Q(s,a) \leftarrow Q(s,a) + \alpha \delta E(s,a)$
            If $A' = A^*$, then $E(s,a) \leftarrow \gamma \lambda E(s,a)$
                            else $E(s,a) \leftarrow 0$
        $S \leftarrow S'$; $A \leftarrow A'$
    until S is terminal

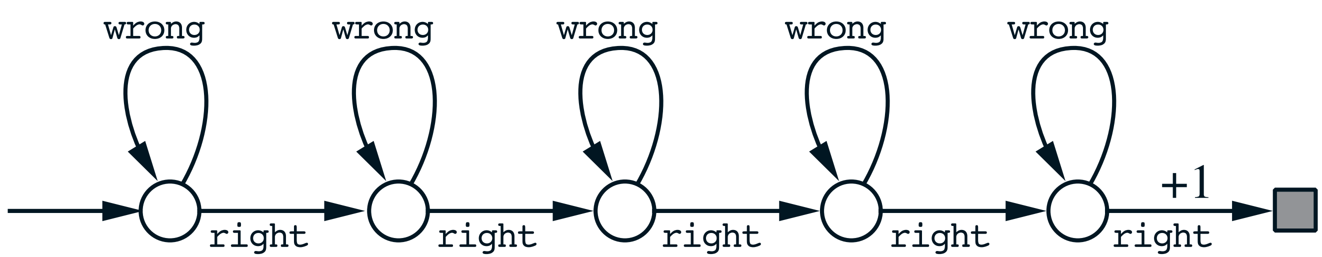
Replacing Traces Example

- Same 19 state random walk task as before
- Replacing traces perform better than accumulating traces over more values of λ



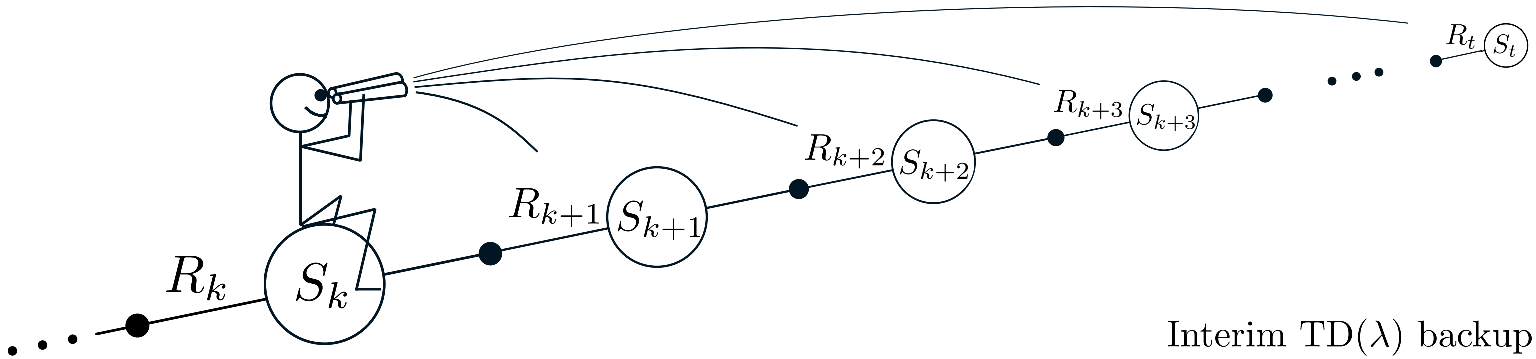
Why Replacing Traces?

- Replacing traces can significantly speed learning
- They can make the system perform well for a broader set of parameters
- Accumulating traces can do poorly on certain types of tasks

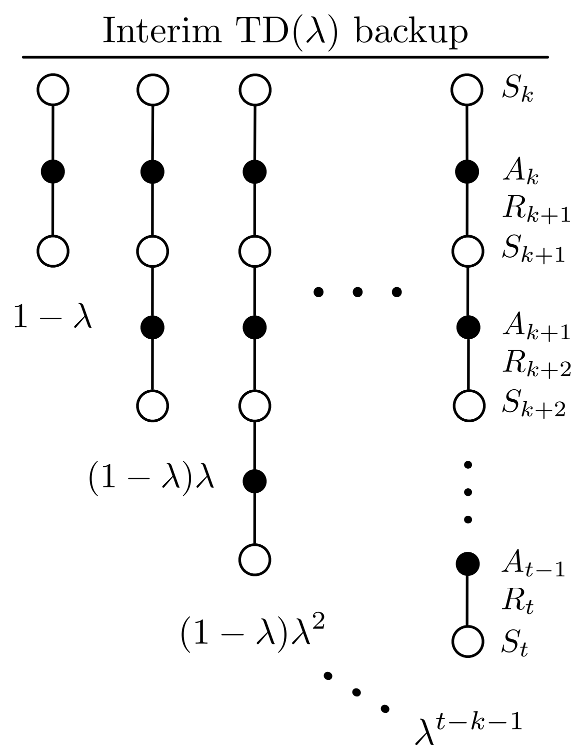


Why is this task particularly onerous for accumulating traces?

Interim TD(λ) Forward View



- At each time t, you can only see the data up to time t
  - so you <u>must bootstrap</u> at time t
- However you can go back and <u>redo all previous updates</u> at times $k < t$
- TD(λ) is equivalent to this
  - exactly under off-line updating
  - approximately under online



True Online TD(λ)

- A new algorithm that more truly achieves the goals of TD(λ) under online updating
  - achieves the interim TD(λ) forward view *exactly*, even under online updating, for any λ, ***γ***
- Not restricted to episodic problems
- Extends immediately to function approximation
- Appears to perform better than both accumulating and replacing traces ("*enhanced*" traces)
- Tabular version:

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

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

$$V_{t+1}(s) = V_t(s) + \alpha\delta_t E_t(s) + (\text{if } s = S_t)\ \alpha(V_{t-1}(S_t) - V_t(S_t))$$

More Replacing Traces

- Off-line replacing trace TD(1) is identical to first-visit MC

- Extension to action-values:
  - When you revisit a state, what should you do with the traces for the other actions?
  - Perhaps you should set them to zero:

$$E_t(s,a) = \begin{cases} 1 & \text{if } s = S_t \text{ and } a = A_t; \\ 0 & \text{if } s = S_t \text{ and } a \neq A_t; \\ \gamma\lambda E_{t-1}(s,a) & \text{if } s \neq S_t. \end{cases} \qquad \text{for all } s, a$$

  - But it is not clear that this is a good idea in all

Implementation Issues with Traces

- Could require much more computation
  - But most eligibility traces are VERY close to zero
  - Really only need to update those
- In practice increases computation by only a small multiple

Variable λ

- Can generalize to variable λ

$$E_t(s) = \begin{cases} \gamma \lambda_t E_{t-1}(s) & \text{if } s \neq S_t \\ \gamma \lambda_t E_{t-1}(s) + 1 & \text{if } s = S_t \end{cases}$$

- Here λ is a function of time
  - Could define

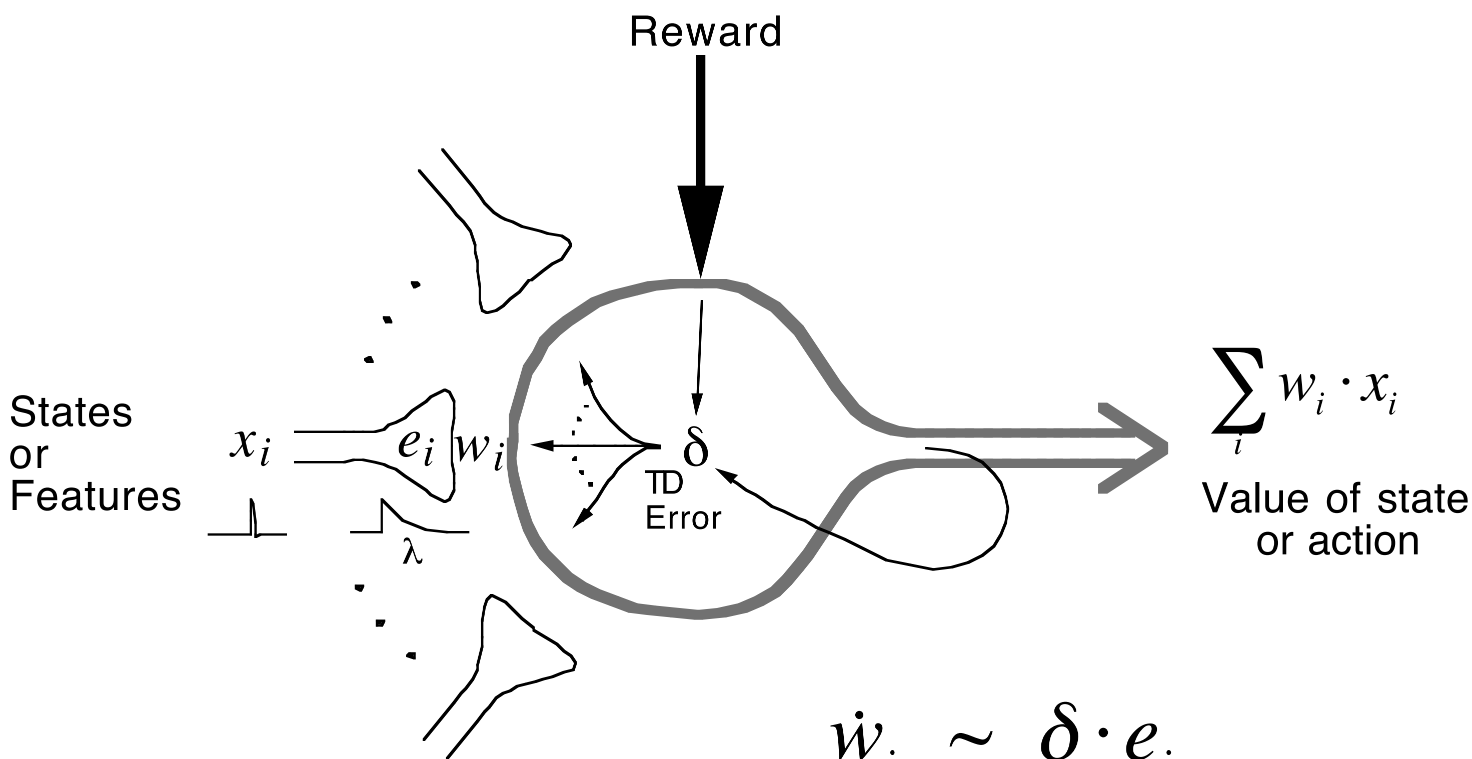
$$\lambda_t = \lambda(s_t) \text{ or } \lambda_t = \lambda^{t/\tau}$$

Conclusions regarding Eligibility Traces

- Provide an efficient, incremental way to combine Monte Carlo (MC) and temporal-difference (TD) learning methods
  - Includes advantages of MC (can deal with lack of Markov property)
  - Includes advantages of TD (using TD error, bootstrapping)
- Can achieve MC behavior even on non-episodic problems
- Can significantly speed learning
- Extends to control in on-policy (Sarsa(λ)) and semi-off-policy (Q(λ)) forms
- Three varieties: *accumulating*, *replacing*, and new

- questions?

TD(λ) algorithm/model/neuron



$$\dot{w}_i \sim \delta \cdot e_i$$