

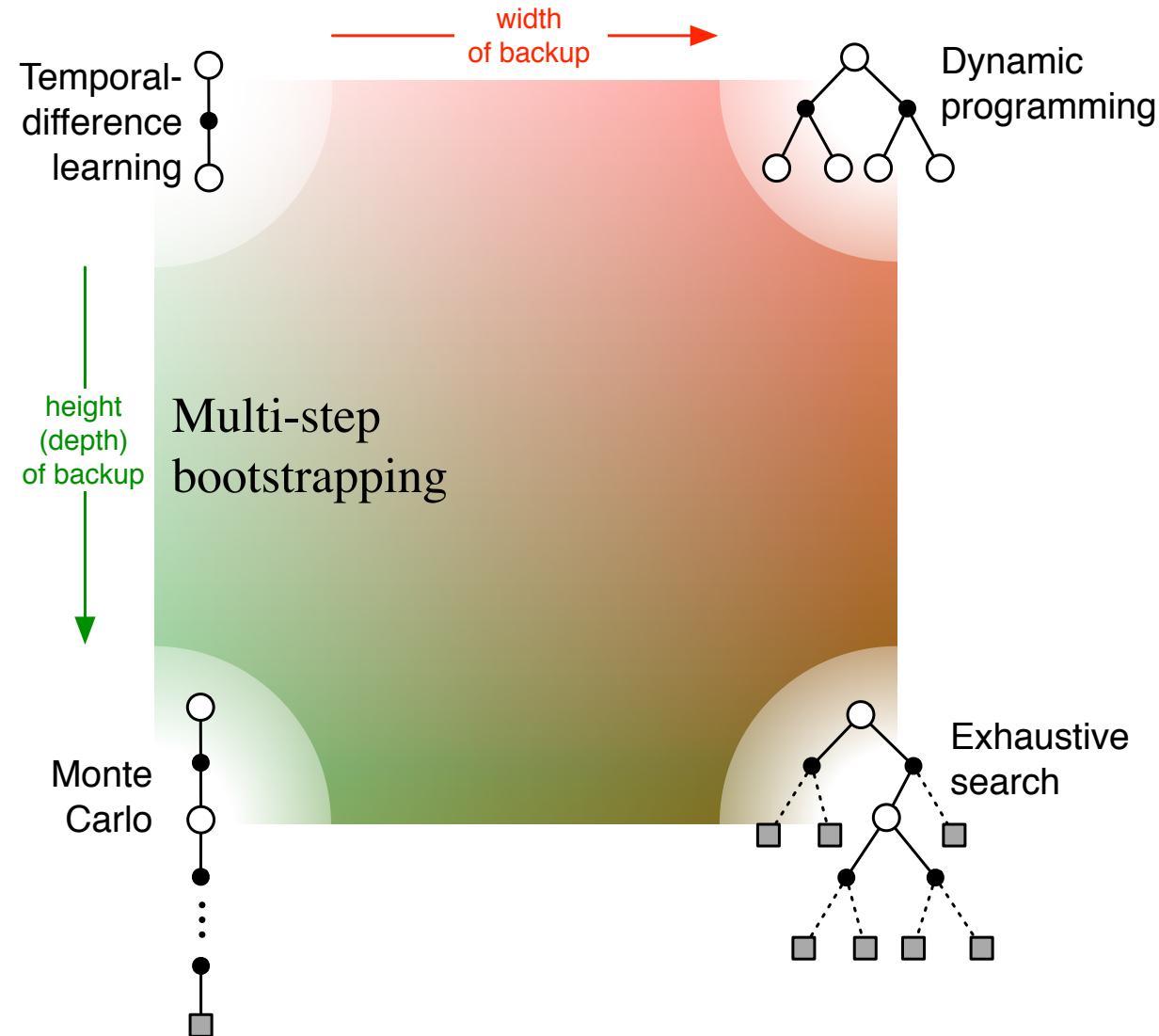
# **Eligibility Traces**

Chapter 12

# Eligibility traces are

- Another way of interpolating between MC and TD methods
- A way of implementing *compound  $\lambda$ -return* targets
- A basic mechanistic idea — a short-term, fading memory
- A new style of algorithm development/analysis
  - the forward-view  $\Leftrightarrow$  backward-view transformation
  - Forward view:  
conceptually simple — good for theory, intuition
  - Backward view:  
computationally congenial implementation of the f. view

# Unified View



# Recall $n$ -step targets

- For example, in the episodic case,  
with linear function approximation:
  - 2-step target:

$$G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 \mathbf{w}_{t+1}^\top \mathbf{x}_{t+2}$$

- $n$ -step target:

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \mathbf{w}_{t+n-1}^\top \mathbf{x}_{t+n}$$

with  $G_{t:t+n} \doteq G_t$  if  $t+n \geq T$

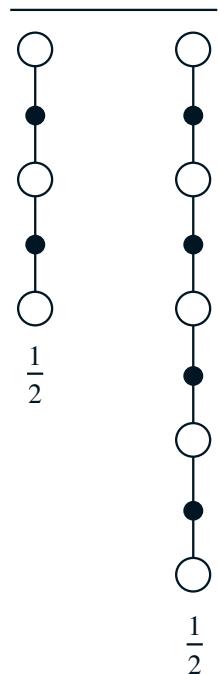
Any set of update targets can be *averaged* to produce new *compound* update targets

- For example, half a 2-step plus half a 4-step

$$U_t = \frac{1}{2}G_{t:t+2} + \frac{1}{2}G_{t:t+4}$$

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

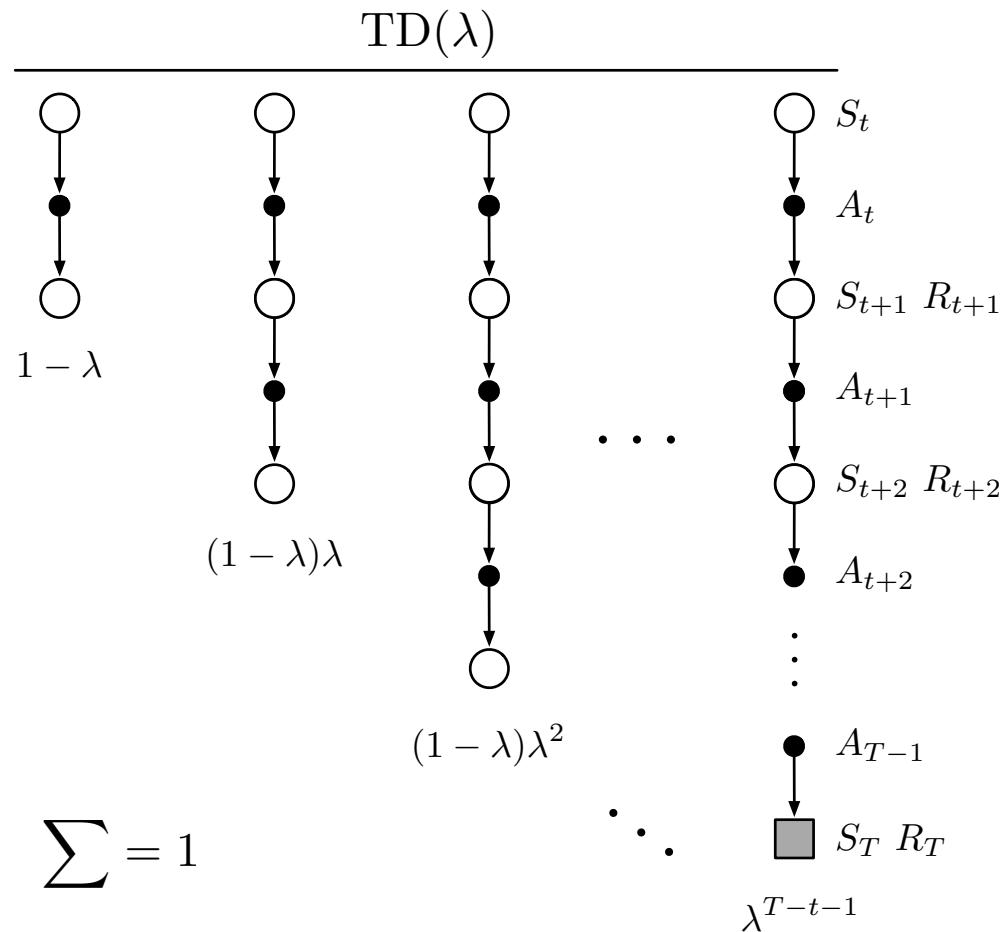
A *compound* backup



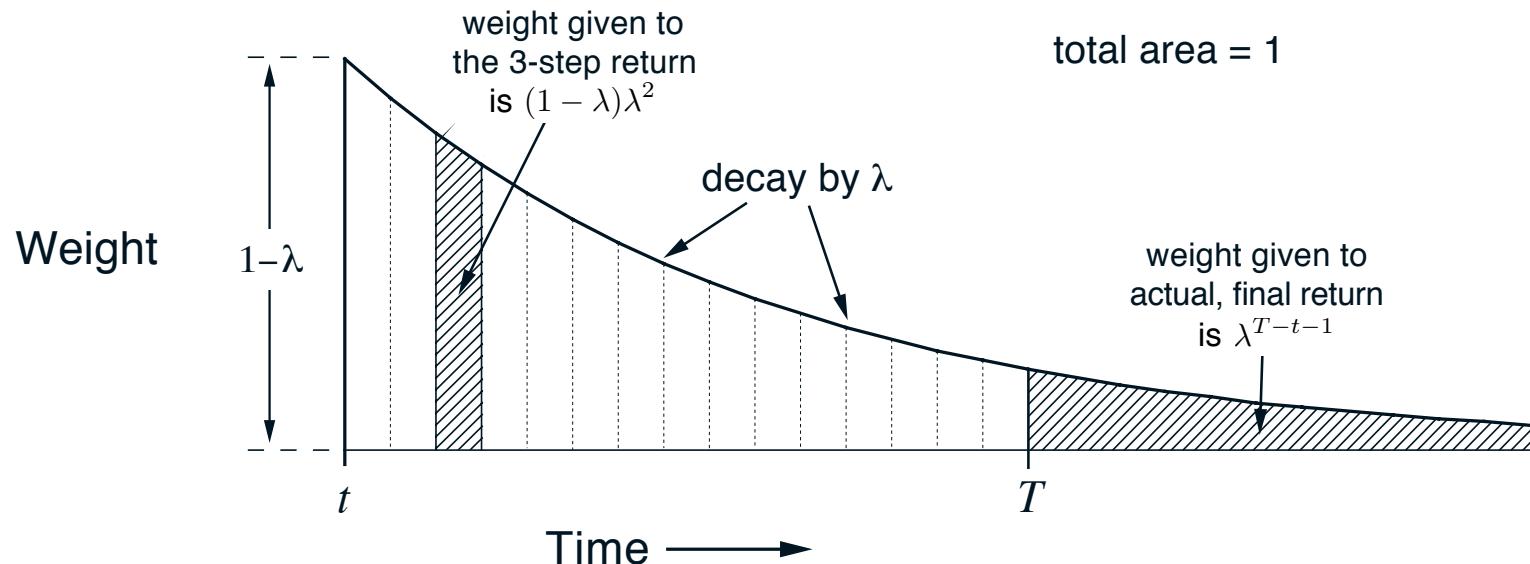
# The $\lambda$ -return is a compound update target

- The  $\lambda$ -return a target that averages all  $n$ -step targets
  - each weighted by  $\lambda^{n-1}$

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



# $\lambda$ -return Weighting Function



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

# Relation to TD(0) and MC

---

- The  $\lambda$ -return can be rewritten as:

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

- If  $\lambda = 1$ , you get the MC target:

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

- If  $\lambda = 0$ , you get the TD(0) target:

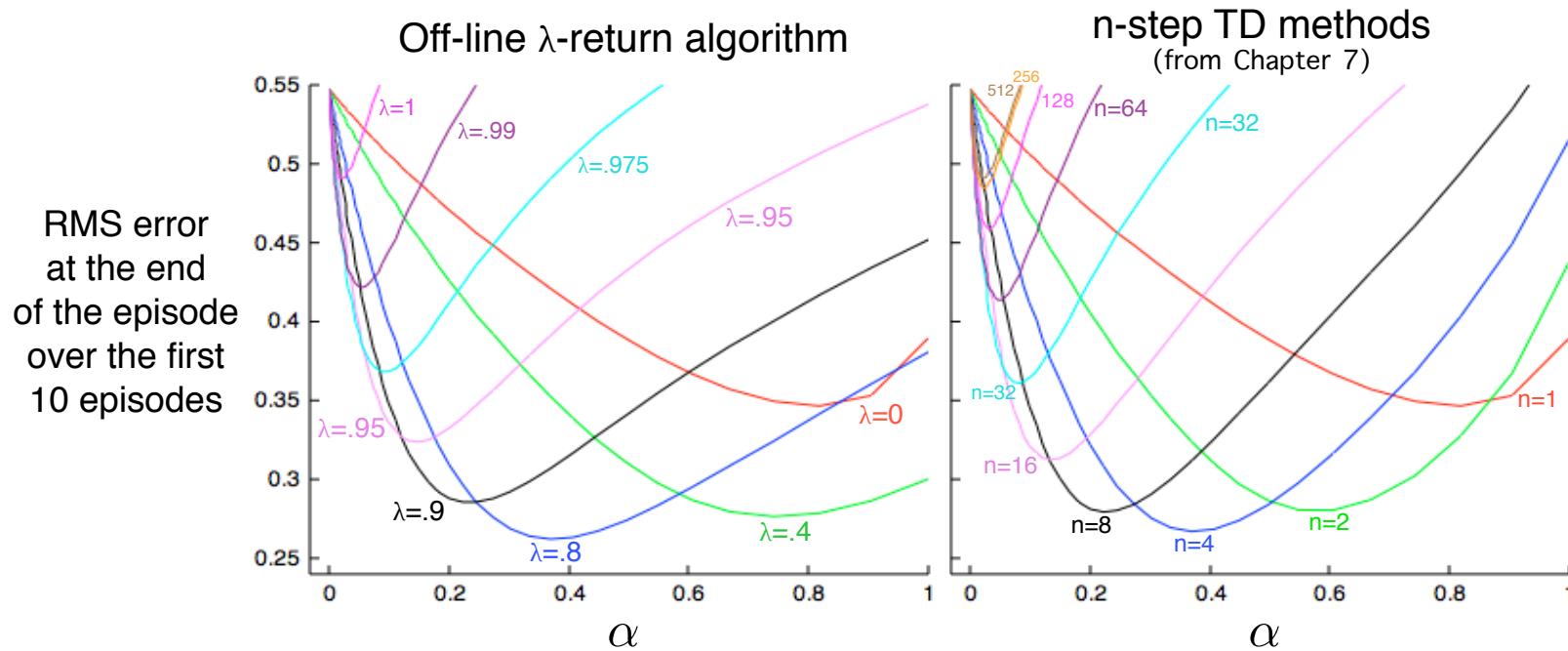
$$G_t^\lambda = (1 - 0) \sum_{n=1}^{T-t-1} 0^{n-1} G_{t:t+n} + 0^{T-t-1} G_t = G_{t:t+1}$$

## The off-line $\lambda$ -return “algorithm”

- Wait until the end of the episode (offline)
- Then go back over the time steps, updating

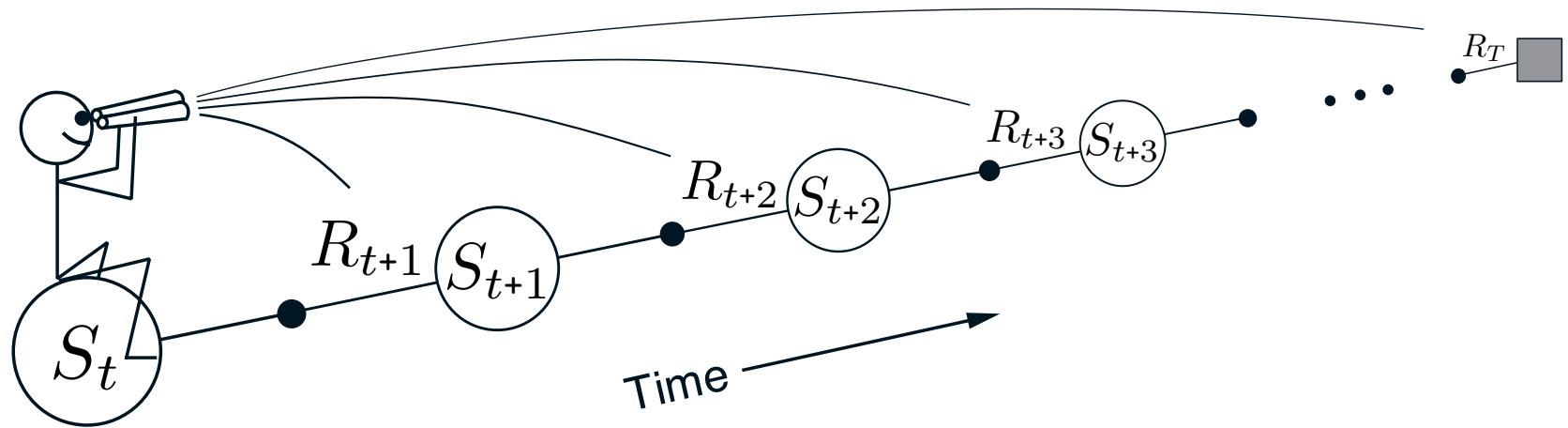
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[ G_t^\lambda - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t), \quad t = 0, \dots, T-1$$

The  $\lambda$ -return alg performs similarly to  $n$ -step algs on the 19-state random walk (Tabular)

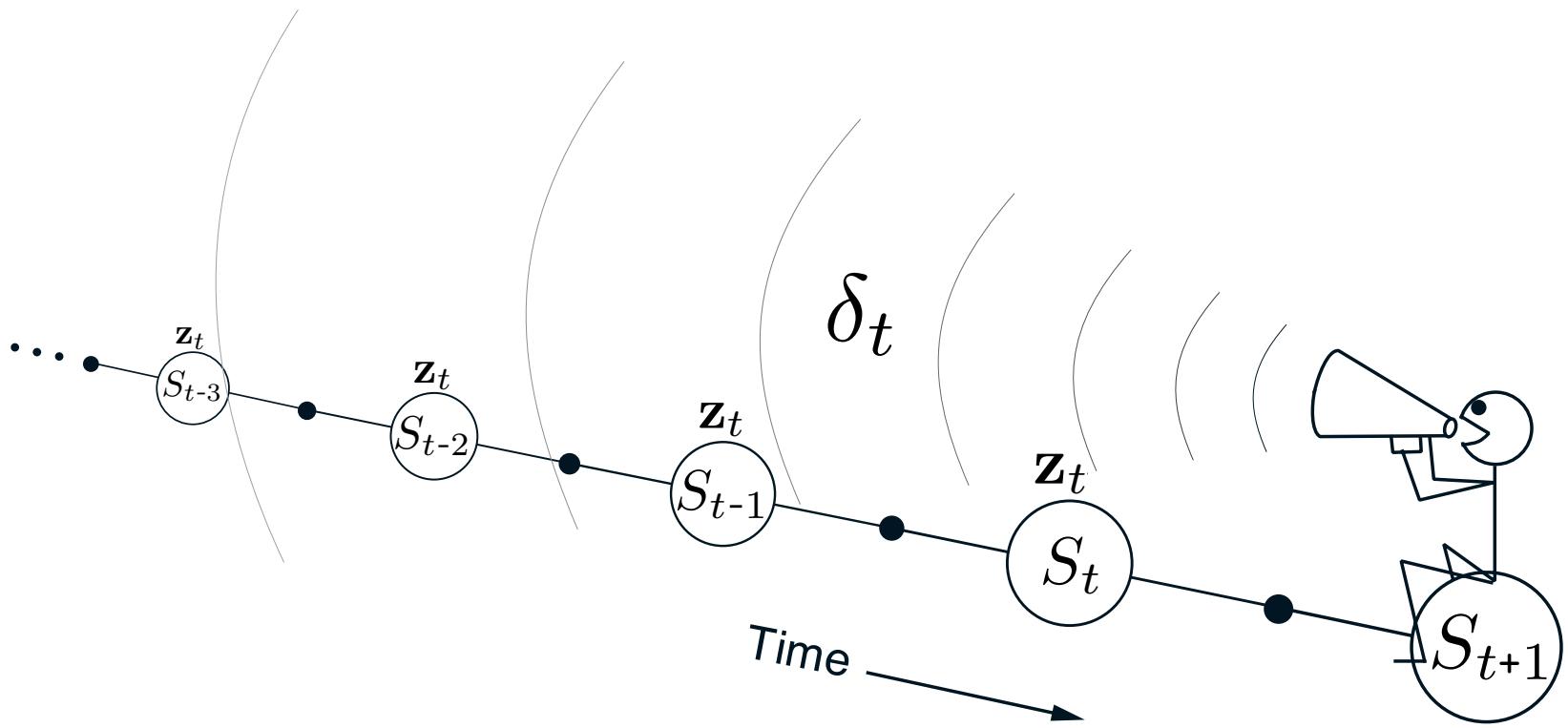


Intermediate  $\lambda$  is best (just like intermediate  $n$  is best)  
 $\lambda$ -return slightly better than  $n$ -step

The forward view looks forward from the state being updated to future states and rewards



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 Semi-gradient TD( $\lambda$ ) algorithm

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t$$

$$\delta_t \doteq R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)$$

$$\mathbf{z}_{-1} \doteq \mathbf{0},$$

$$\mathbf{z}_t \doteq \gamma \lambda \mathbf{z}_{t-1} + \nabla \hat{v}(S_t, \mathbf{w}_t)$$

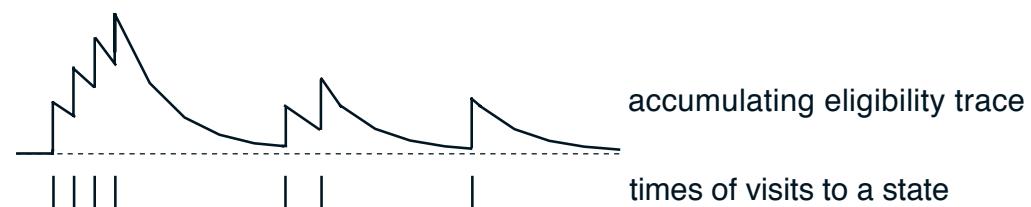
New error bound:

$$\overline{\text{VE}}(\mathbf{w}_\infty) \leq \frac{1 - \gamma \lambda}{1 - \gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w})$$

# Eligibility traces (mechanism)

- The forward view was for theory
- The backward view is for *mechanism*
- New memory vector called *eligibility trace*  $\mathbf{z}_t \in \mathbb{R}^d$ 
  - On each step, decay each component by  $\gamma\lambda$  and increment the trace for the current state by 1
  - *Accumulating trace*

$$\mathbf{z}_{-1} \doteq \mathbf{0}, \\ \mathbf{z}_t \doteq \gamma\lambda \mathbf{z}_{t-1} + \nabla \hat{v}(S_t, \mathbf{w}_t)$$



## Semi-gradient TD( $\lambda$ ) for estimating $\hat{v} \approx v_\pi$

Input: the policy  $\pi$  to be evaluated

Input: a differentiable function  $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \rightarrow \mathbb{R}$  such that  $\hat{v}(\text{terminal}, \cdot) = 0$

Initialize value-function weights  $\mathbf{w}$  arbitrarily (e.g.,  $\mathbf{w} = \mathbf{0}$ )

Repeat (for each episode):

Initialize  $S$

$\mathbf{z} \leftarrow \mathbf{0}$  (a  $d$ -dimensional vector)

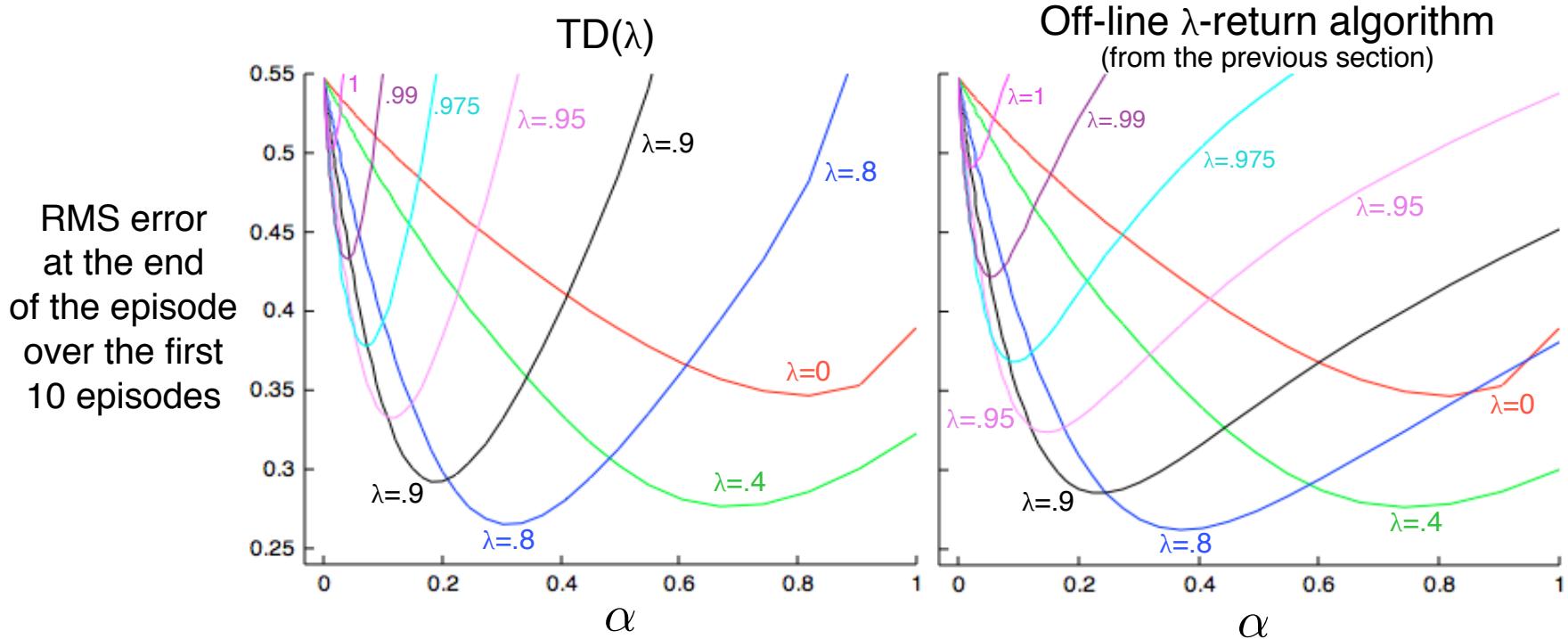
Repeat (for each step of episode):

- . Choose  $A \sim \pi(\cdot | S)$
  - . Take action  $A$ , observe  $R, S'$
  - .  $\mathbf{z} \leftarrow \gamma \lambda \mathbf{z} + \nabla \hat{v}(S, \mathbf{w})$
  - .  $\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
  - .  $\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \mathbf{z}$
  - .  $S \leftarrow S'$

until  $S'$  is terminal

$\text{TD}(\lambda)$  performs similarly to offline  $\lambda$ -return alg.  
but slightly worse, particularly at high  $\alpha$

Tabular 19-state random walk task

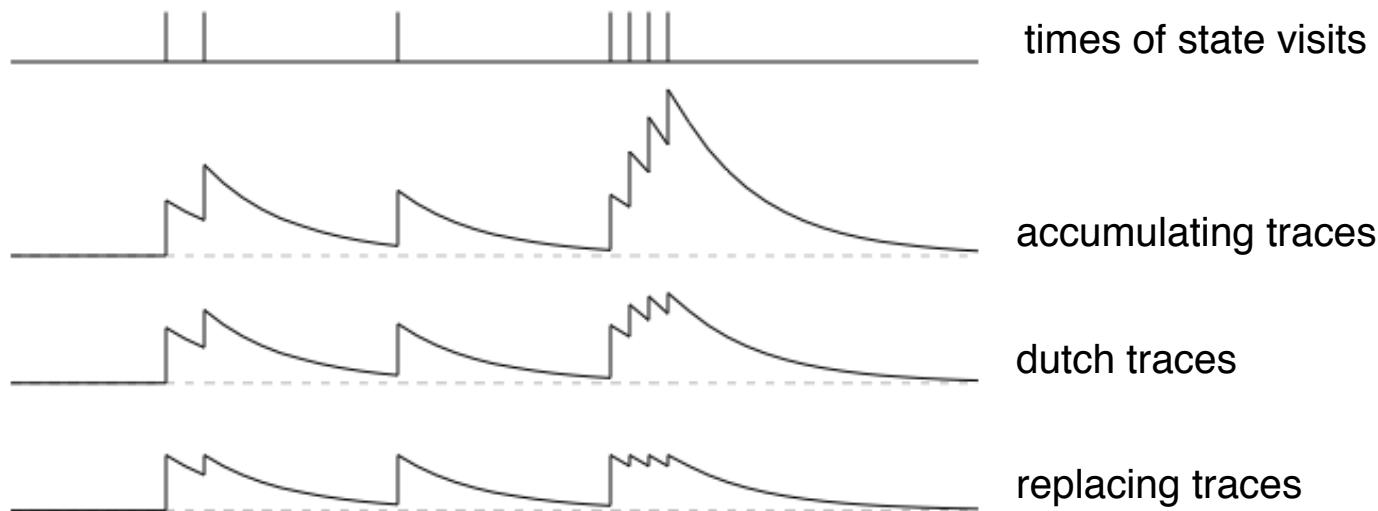


Can we do better? Can we update online?

# Accumulating, Dutch, and Replacing Traces

---

- All traces fade the same:
- But increment differently!



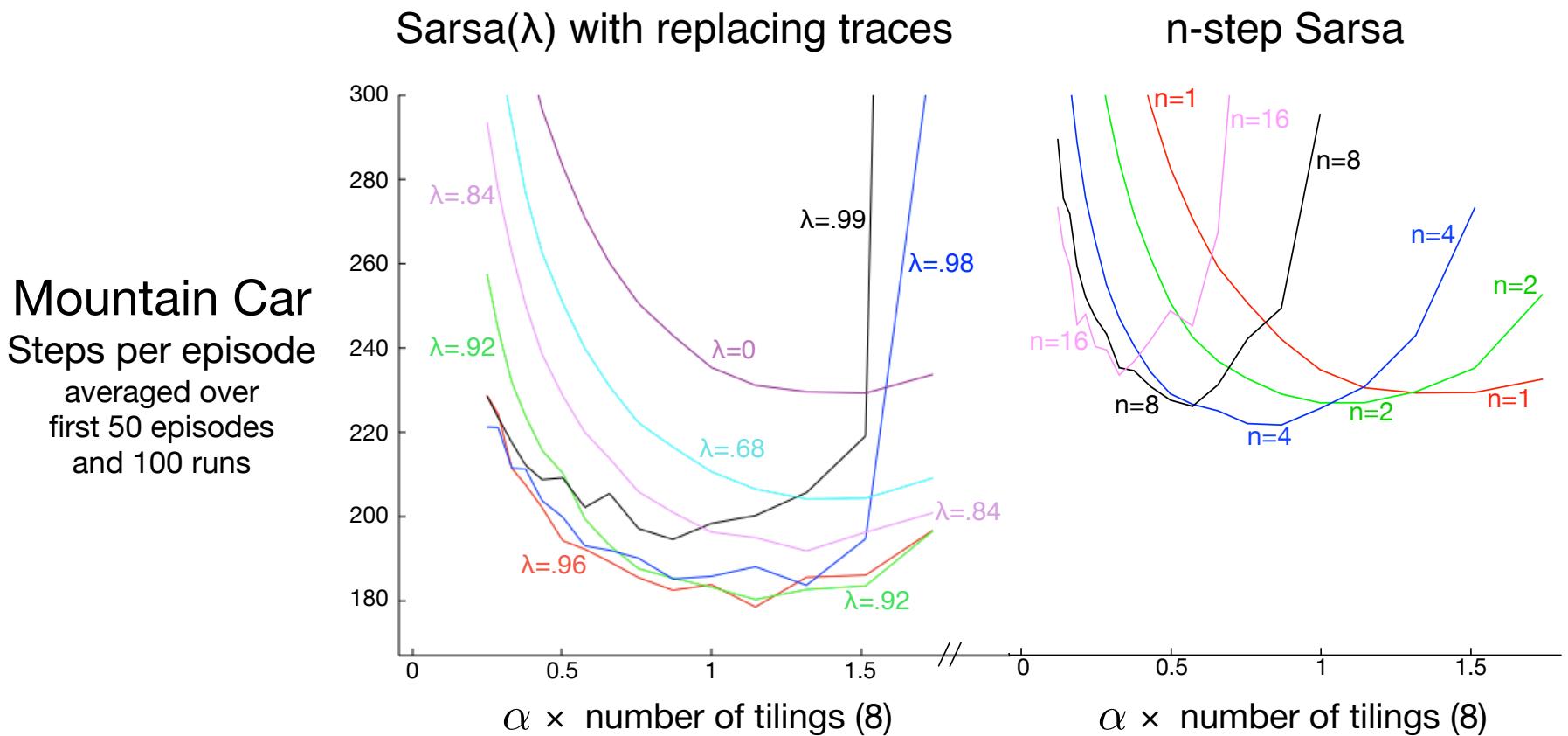
# Traces for control — Sarsa( $\lambda$ ) (on-policy)

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t$$

$$\delta_t \doteq R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)$$

$$\begin{aligned}\mathbf{z}_{-1} &\doteq \mathbf{0}, \\ \mathbf{z}_t &\doteq \gamma \lambda \mathbf{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)\end{aligned}$$

# Sarsa( $\lambda$ ) is better than n-step Sarsa on Mountain Car with tile-coding linear function approximation



# Conclusions regarding Eligibility Traces

---

- Provide an efficient, incremental way to combine MC and TD
  - Includes advantages of MC (better when non-Markov)
  - Includes advantages of TD (faster, comp. congenial)
- True online  $\text{TD}(\lambda)$  is new and best
  - Is exactly equivalent to online  $\lambda$ -return algorithm
- Three varieties of traces: accumulating, dutch, (replacing)
- Traces for prediction and on-policy control
- Trace methods often perform better than  $n$ -step methods
- Traces do have a small cost in computation ( $\approx \times 2$ )

## The simplest example of deriving a backward view from a forward view

- Monte Carlo learning of a final target
- Will derive dutch traces
- Showing the dutch traces really are not about TD
- They are about efficiently implementing online algs

# The Problem:

Predict final target  $G$  with linear function approximation

	episode						next episode		
Time	0	1	2	...	T-1	T	0	1	2
Data	$\mathbf{x}_0$	$\mathbf{x}_1$	$\mathbf{x}_2$	...	$\mathbf{x}_{T-1}$	$G$			
Weights	$\mathbf{w}_0$	$\mathbf{w}_0$	$\mathbf{w}_0$	...	$\mathbf{w}_0$	$\mathbf{w}_T$	$\mathbf{w}_T$	$\mathbf{w}_T$	$\mathbf{w}_T$
Predictions	$\mathbf{w}_0^\top \mathbf{x}_0$	$\mathbf{w}_0^\top \mathbf{x}_1$	$\mathbf{w}_0^\top \mathbf{x}_2$	...	$\mathbf{w}_0^\top \mathbf{x}_{T-1}$				
	$\approx G$								

$$\text{MC: } \mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha_t (G - \mathbf{w}_t^\top \mathbf{x}_t) \mathbf{x}_t$$

step size

all done at time  $T$

# Computational goals

Computation per step (including memory) must be

1. *Constant*. (non-increasing with number of episodes)
2. *Proportionate*. (proportional to number of weights, or  $O(n)$ )
3. *Independent of span*. (not increasing with episode length)  
In general, the *predictive span* is the number of steps between making a prediction and observing the outcome

MC:  $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha [G - \mathbf{w}_t^\top \mathbf{x}_t] \mathbf{x}_t,$        $t = 0, \dots, T - 1$

all done at time  $T$

step size

What is the span?  $T$   
Is MC indep of span? No

# Computational goals

Computation per step (including memory) must be

1. *Constant*. (non-increasing with number of episodes)
2. *Proportionate*. (proportional to number of weights, or  $O(n)$ )
3. *Independent of span*. (not increasing with episode length)  
In general, the *predictive span* is the number of steps between making a prediction and observing the outcome

$$\text{MC: } \mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha [G - \mathbf{w}_t^\top \mathbf{x}_t] \mathbf{x}_t, \quad t = 0, \dots, T-1$$

step size

all done at time  $T$

Computation and memory needed at step  $T$  increases with  $T \Rightarrow$  not IoS

# Final Result

Given:

$$\mathbf{w}_0 \quad \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T-1} \quad G$$

MC algorithm:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(G - \mathbf{w}_t^\top \mathbf{x}_t) \mathbf{x}_t, \quad t = 0, \dots, T-1$$

Equivalent independent-of-span algorithm:

$$\begin{aligned} \mathbf{w}_T &\doteq \mathbf{a}_{T-1} + G \mathbf{z}_{T-1}, & \mathbf{a}_t \in \mathbb{R}^d, \quad \mathbf{z}_t \in \mathbb{R}^d \\ \mathbf{a}_0 &\doteq \mathbf{w}_0, \text{ then } \mathbf{a}_t \doteq \mathbf{a}_{t-1} - \alpha_t \mathbf{x}_t \mathbf{x}_t^\top \mathbf{a}_{t-1}, & t = 1, \dots, T-1 \\ \mathbf{z}_0 &\doteq \alpha_0 \mathbf{x}_0, \text{ then } \mathbf{z}_t \doteq \mathbf{z}_{t-1} - \alpha_t \mathbf{x}_t \mathbf{x}_t^\top \mathbf{z}_{t-1} + \alpha_t \mathbf{x}_t, & t = 1, \dots, T-1 \end{aligned}$$

Proved:

$$\mathbf{w}_T = \mathbf{w}_T \quad (\text{the final weights of both algorithms are the same})$$

$$\text{MC: } \mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(G - \mathbf{w}_t^\top \mathbf{x}_t) \mathbf{x}_t, \quad t = 0, \dots, T-1$$

$$\begin{aligned}\mathbf{w}_T &= \mathbf{w}_{T-1} + \alpha \left( G - \mathbf{w}_{T-1}^\top \mathbf{x}_{T-1} \right) \mathbf{x}_{T-1} \\ &= \mathbf{w}_{T-1} + \alpha \mathbf{x}_{T-1} \left( -\mathbf{x}_{T-1}^\top \mathbf{w}_{T-1} \right) + \alpha G \mathbf{x}_{T-1} \\ &= \left( \mathbf{I} - \alpha \mathbf{x}_{T-1} \mathbf{x}_{T-1}^\top \right) \mathbf{w}_{T-1} + \alpha G \mathbf{x}_{T-1} \\ &= \mathbf{F}_{T-1} \mathbf{w}_{T-1} + \alpha G \mathbf{x}_{T-1}\end{aligned}$$

where  $\mathbf{F}_t \doteq \mathbf{I} - \alpha \mathbf{x}_t \mathbf{x}_t^\top$  is a *forgetting*, or *fading*, matrix. Now, recursing,

$$\begin{aligned}&= \mathbf{F}_{T-1} (\mathbf{F}_{T-2} \mathbf{w}_{T-2} + \alpha G \mathbf{x}_{T-2}) + \alpha G \mathbf{x}_{T-1} \\ &= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{w}_{T-2} + \alpha G (\mathbf{F}_{T-1} \mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\ &= \mathbf{F}_{T-1} \mathbf{F}_{T-2} (\mathbf{F}_{T-3} \mathbf{w}_{T-3} + \alpha G \mathbf{x}_{T-3}) + \alpha G (\mathbf{F}_{T-1} \mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\ &= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{F}_{T-3} \mathbf{w}_{T-3} + \alpha G (\mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{x}_{T-3} + \mathbf{F}_{T-1} \mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\ &\quad \vdots \\ &= \underbrace{\mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_0 \mathbf{w}_0}_{\mathbf{a}_{T-1}} + \underbrace{\alpha G \sum_{k=0}^{T-1} \mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_{k+1} \mathbf{x}_k}_{\mathbf{z}_{T-1}} \\ &= \mathbf{a}_{T-1} + \alpha G \mathbf{z}_{T-1}, \quad \text{auxiliary short-term-memory vectors } \mathbf{a}_t \in \mathbb{R}^d, \mathbf{z}_t \in \mathbb{R}^d\end{aligned}$$

$$\begin{aligned}
&= \underbrace{\mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_0 \mathbf{w}_0}_{\mathbf{a}_{T-1}} + \underbrace{\alpha G \sum_{k=0}^{T-1} \mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_{k+1} \mathbf{x}_k}_{\mathbf{z}_{T-1}} \\
&= \mathbf{a}_{T-1} + \alpha G \mathbf{z}_{T-1},
\end{aligned}$$

$$\begin{aligned}
&= \underbrace{\mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_0 \mathbf{w}_0}_{\mathbf{a}_{T-1}} + \underbrace{\alpha G \sum_{k=0}^{T-1} \mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_{k+1} \mathbf{x}_k}_{\mathbf{z}_{T-1}} \\
&= \mathbf{a}_{T-1} + \alpha G \mathbf{z}_{T-1},
\end{aligned}$$

$$\begin{aligned}
\mathbf{z}_t &\doteq \sum_{k=0}^t \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k, & 1 \leq t < T \\
&= \sum_{k=0}^{t-1} \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k + \mathbf{x}_t \\
&= \mathbf{F}_t \sum_{k=0}^{t-1} \mathbf{F}_{t-1} \mathbf{F}_{t-2} \cdots \mathbf{F}_{k+1} \mathbf{x}_k + \mathbf{x}_t \\
&= \mathbf{F}_t \mathbf{z}_{t-1} + \mathbf{x}_t \\
&= (\mathbf{I} - \alpha \mathbf{x}_t \mathbf{x}_t^\top) \mathbf{z}_{t-1} + \mathbf{x}_t \\
&= \mathbf{z}_{t-1} - \alpha \mathbf{x}_t \mathbf{x}_t^\top \mathbf{z}_{t-1} + \mathbf{x}_t \\
&= \mathbf{z}_{t-1} - \alpha (\mathbf{z}_{t-1}^\top \mathbf{x}_t) \mathbf{x}_t + \mathbf{x}_t \\
&= \mathbf{z}_{t-1} + (1 - \alpha \mathbf{z}_{t-1}^\top \mathbf{x}_t) \mathbf{x}_t,
\end{aligned}$$

$$\mathbf{a}_t \doteq \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_0 \mathbf{w}_0 = \mathbf{F}_t \mathbf{a}_{t-1} = \mathbf{a}_{t-1} - \alpha \mathbf{x}_t \mathbf{x}_t^\top \mathbf{a}_{t-1}, \quad 1 \leq t < T$$

# Final Result

Given:

$$\mathbf{w}_0 \quad \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T-1} \quad G$$

MC algorithm:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(G - \mathbf{w}_t^\top \mathbf{x}_t) \mathbf{x}_t, \quad t = 0, \dots, T-1$$

Equivalent independent-of-span algorithm:

$$\begin{aligned} \mathbf{w}_T &\doteq \mathbf{a}_{T-1} + G \mathbf{z}_{T-1}, & \mathbf{a}_t \in \mathbb{R}^d, \quad \mathbf{z}_t \in \mathbb{R}^d \\ \mathbf{a}_0 &\doteq \mathbf{w}_0, \text{ then } \mathbf{a}_t \doteq \mathbf{a}_{t-1} - \alpha_t \mathbf{x}_t \mathbf{x}_t^\top \mathbf{a}_{t-1}, & t = 1, \dots, T-1 \\ \mathbf{z}_0 &\doteq \alpha_0 \mathbf{x}_0, \text{ then } \mathbf{z}_t \doteq \mathbf{z}_{t-1} - \alpha_t \mathbf{x}_t \mathbf{x}_t^\top \mathbf{z}_{t-1} + \alpha_t \mathbf{x}_t, & t = 1, \dots, T-1 \end{aligned}$$

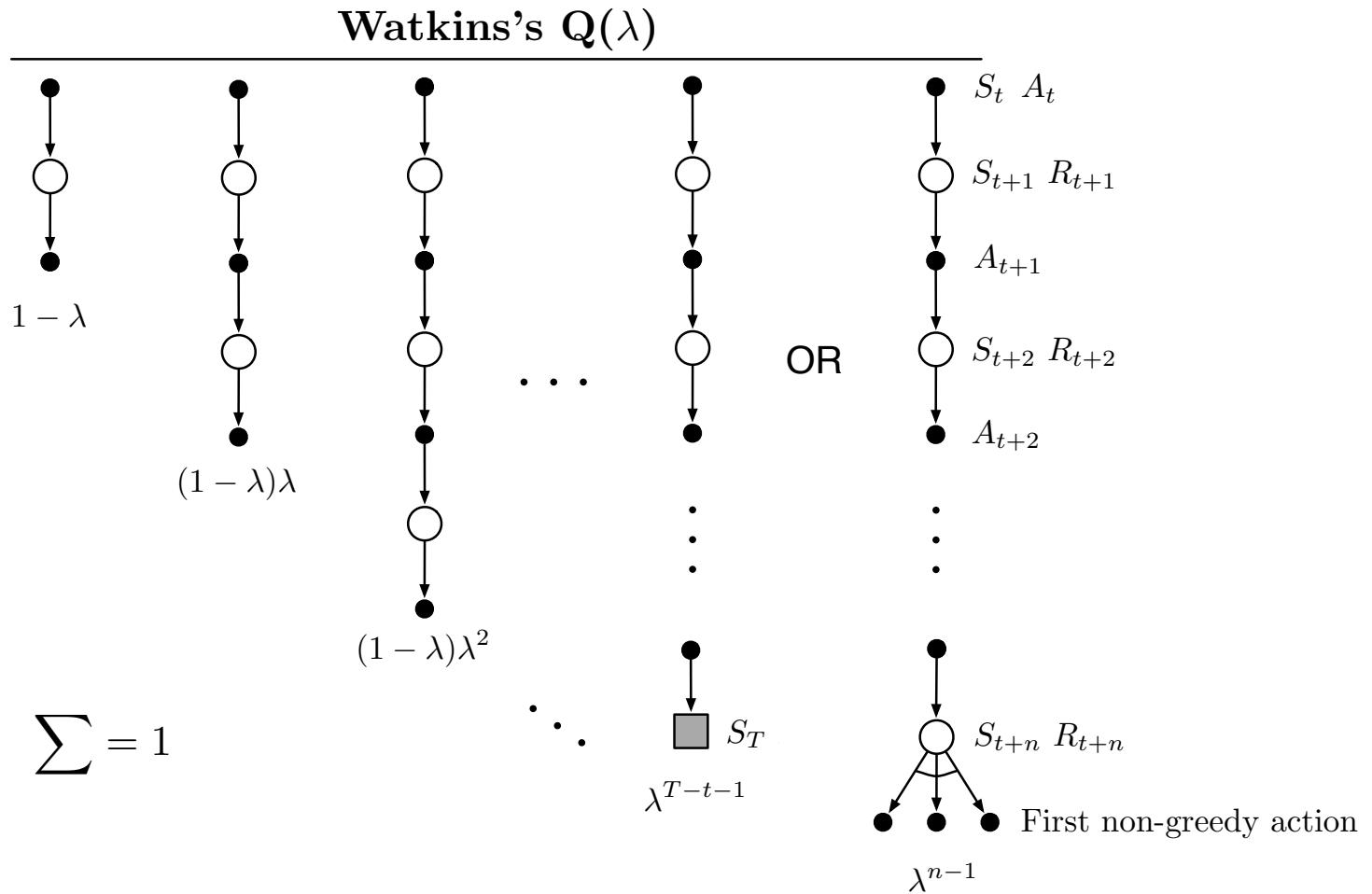
Proved:

$$\mathbf{w}_T = \mathbf{w}_T \quad (\text{the final weights of both algorithms are the same})$$

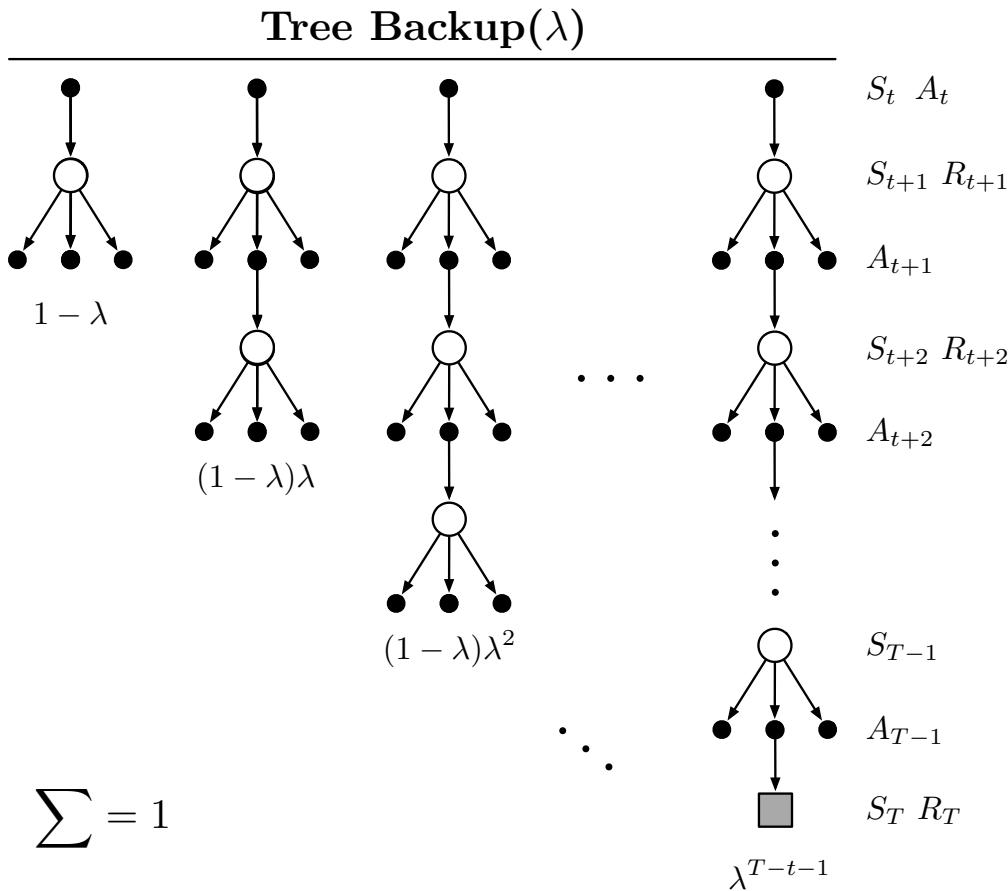
# Conclusions from the forward-backward derivation

- We have derived dutch eligibility traces from an MC update, without any TD learning
- Dutch traces, and in fact all eligibility traces, are not about TD; they are about *efficient multi-step* learning
- We can derive new non-obvious algorithms that are equivalent to obvious algorithms but have better computational properties
- This is a different type of machine-learning result, an *algorithm equivalence*

# Other traces for Q: Original Watkins Q( $\lambda$ )



# Other traces for Q: Tree-Backup( $\lambda$ )



# Other traces for Q: Tree-Backup( $\lambda$ )

## Update Rules

$$\mathbf{z}_t \doteq \gamma_t \lambda_t \pi(A_t | S_t) \mathbf{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t$$

- No importance sampling
- No guarantees of stability when used off-policy with powerful function approximation

# Off-policy Traces with importance sampling

- Learning about an arbitrary policy typically requires the use of importance sampling ratios between the behavior policy and the target policy.

$$\rho_t = \frac{\pi(A_t | S_t)}{b(A_t | S_t)}$$

- We define state based returns, and a forward view update.

$$G_t^{\lambda s} \doteq \rho_t \left( R_{t+1} + \gamma_{t+1} ((1 - \lambda_{t+1}) \hat{v}(S_{t+1}, \mathbf{w}_t) + \lambda_{t+1} G_{t+1}^{\lambda s}) \right) + (1 - \rho_t) \hat{v}(S_t, \mathbf{w}_t)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha (G_t^{\lambda s} - \hat{v}(S_t, \mathbf{w}_t)) \nabla \hat{v}(S_t, \mathbf{w}_t)$$

- After some work (Section 12.9), we get another trace.

$$\mathbf{z}_t \doteq \rho_t (\gamma_t \lambda_t \mathbf{z}_{t-1} + \nabla \hat{v}(S_t, \mathbf{w}_t))$$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t$$

- This is not guaranteed to be stable with strong function approximation, and importance sampling can introduce substantial variance. Can still work in practice.

# Conclusions regarding Eligibility Traces

---

- Provide an efficient, incremental way to combine MC and TD
  - Includes advantages of MC (better when non-Markov)
  - Includes advantages of TD (faster, comp. congenial)
- True online  $\text{TD}(\lambda)$  is new and best
  - Is exactly equivalent to online  $\lambda$ -return algorithm
- There is a true online Sarsa( $\lambda$ )
- Three varieties of traces: accumulating, dutch, (replacing)
- Traces for prediction and on-policy control
- Traces for off-policy control and prediction
- Trace methods often perform better than  $n$ -step methods
- Traces do have a small cost in computation ( $\approx \times 2$ )