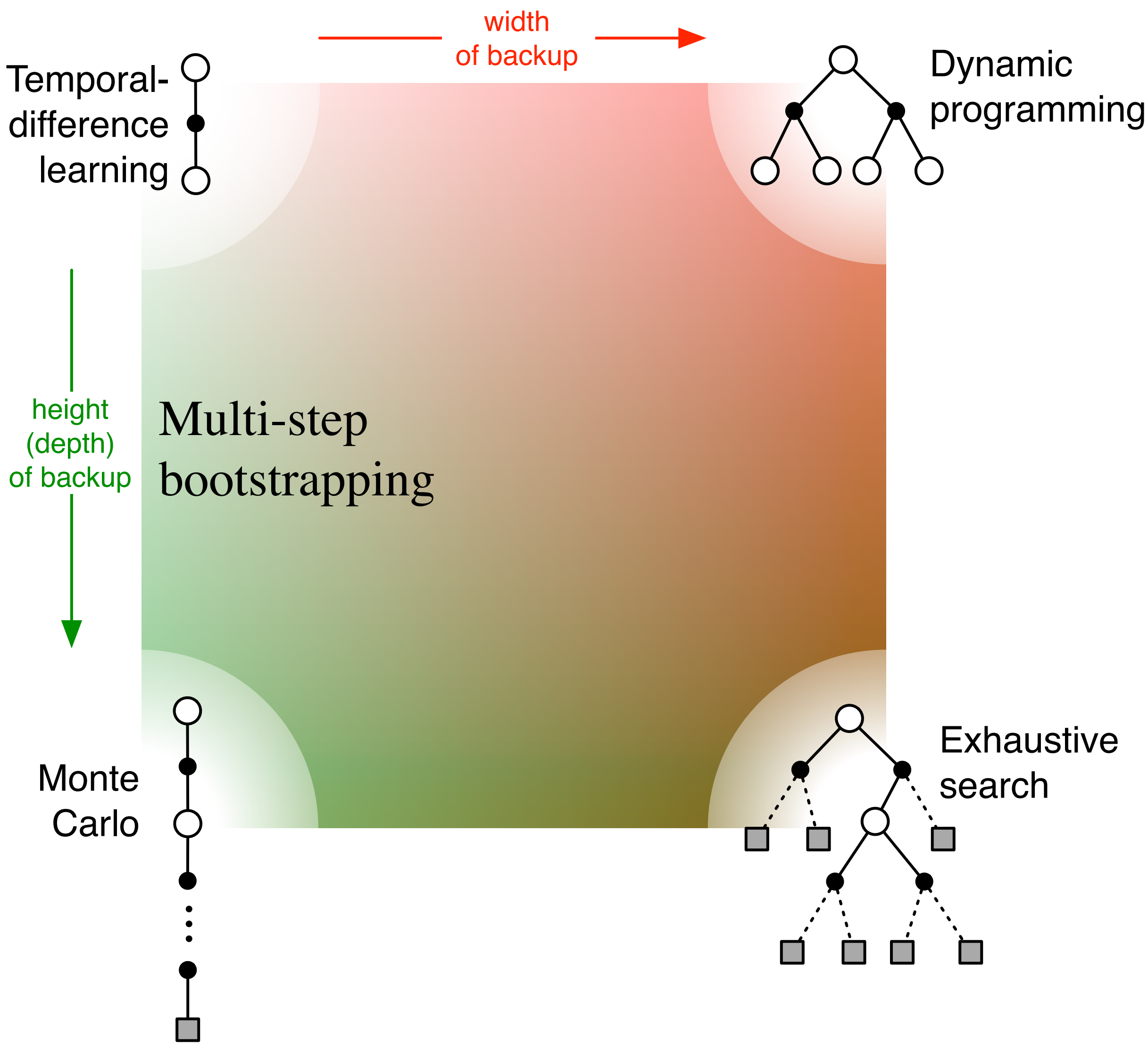
Eligibility Traces

Chapter 12

Eligibility traces are

- Another way of interpolating between MC and TD methods
- A way of implementing *compound λ-return* targets
- A basic mechanistic idea — a short-term, fading memory
- A new style of algorithm development/analysis
  - the forward-view ⇔ 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:

$$G_t^{(2)} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 \boldsymbol{\theta}_{t+1}^\top \boldsymbol{\phi}_{t+2}$$

  - *n*-step target:

$$G_t^{(n)} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \boldsymbol{\theta}_{t+n-1}^\top \boldsymbol{\phi}_{t+n}$$

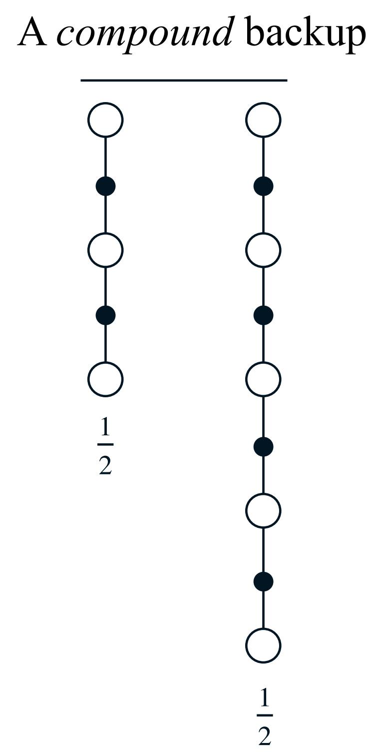
with $G_t^{(n)} \doteq G_t$ if $t + n \geq T$

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

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

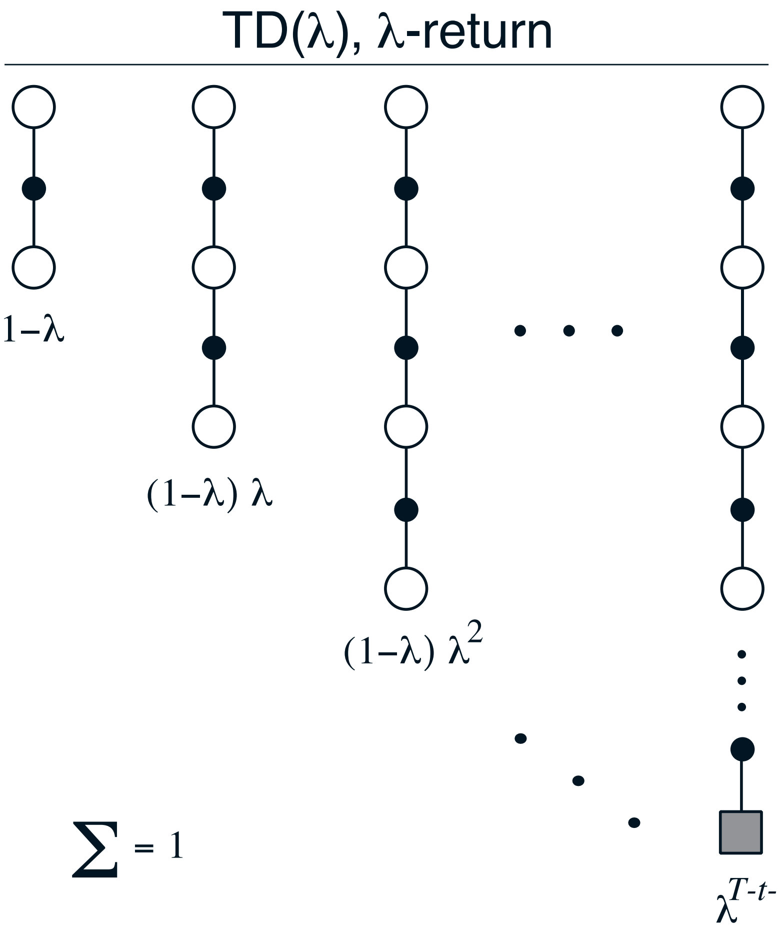
- Called a compound backup
    - Draw each component
    - Label with the weights for that component



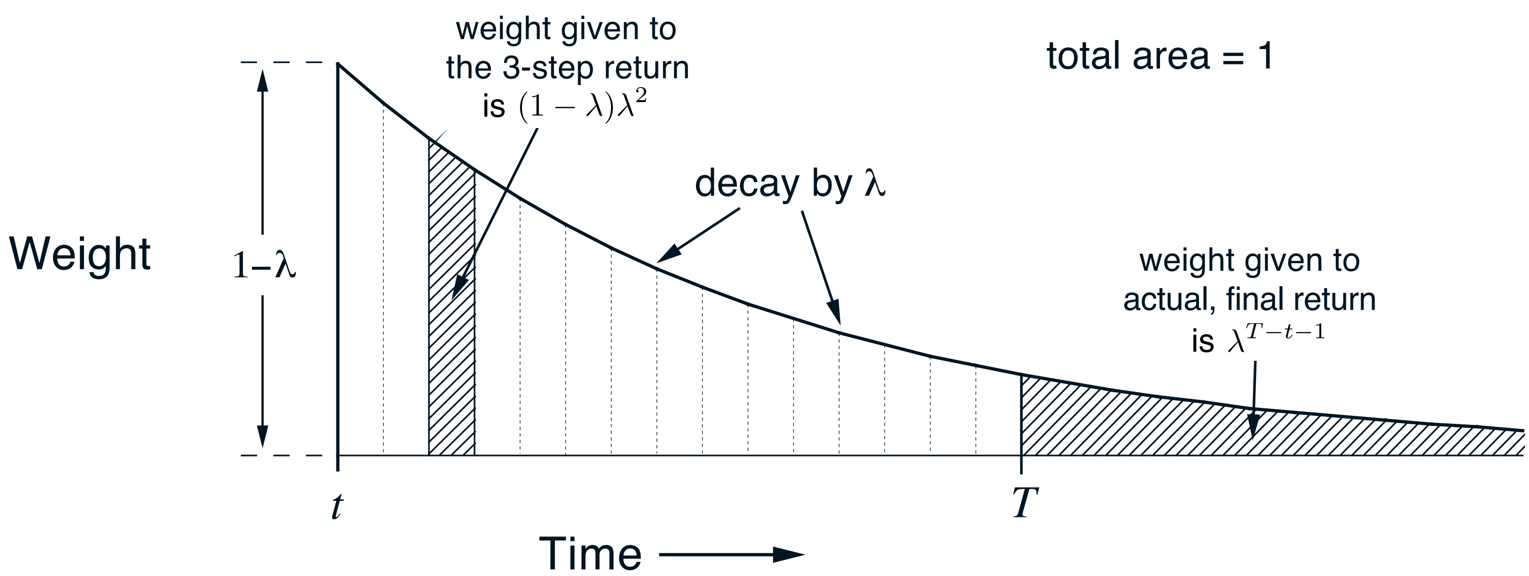
The λ-return is a compound update target

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

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



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 the MC target:

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

- If λ = 0, you get the TD(0) target:

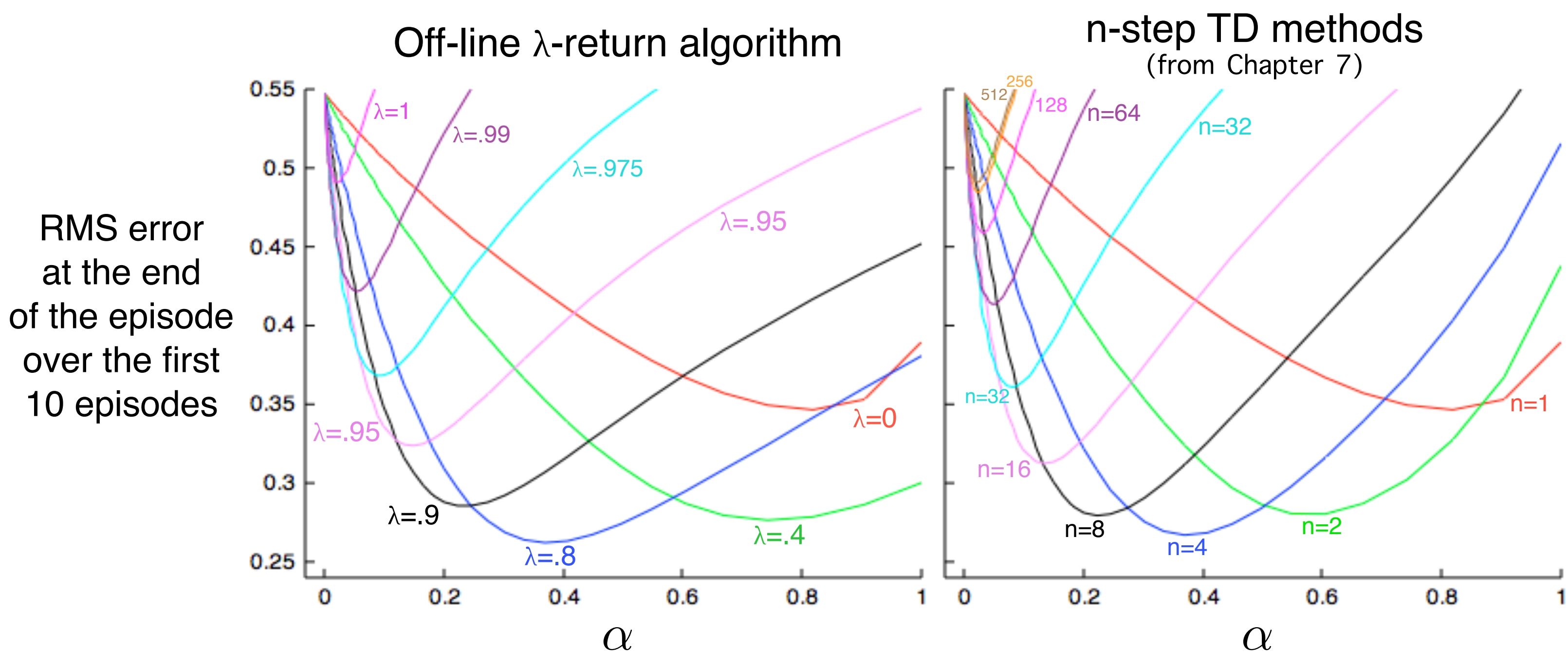
$$G_t^\lambda = (1-0)\sum_{n=1}^{T-t-1}0^{n-1}G_t^{(n)} + 0^{T-t-1}G_t = G_t^{(1)}$$

The off-line λ-return "algorithm"

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

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \Big[G_t^\lambda - \hat{v}(S_t, \boldsymbol{\theta}_t) \Big] \nabla \hat{v}(S_t, \boldsymbol{\theta}_t), \quad t = 0, \ldots, T-1$$

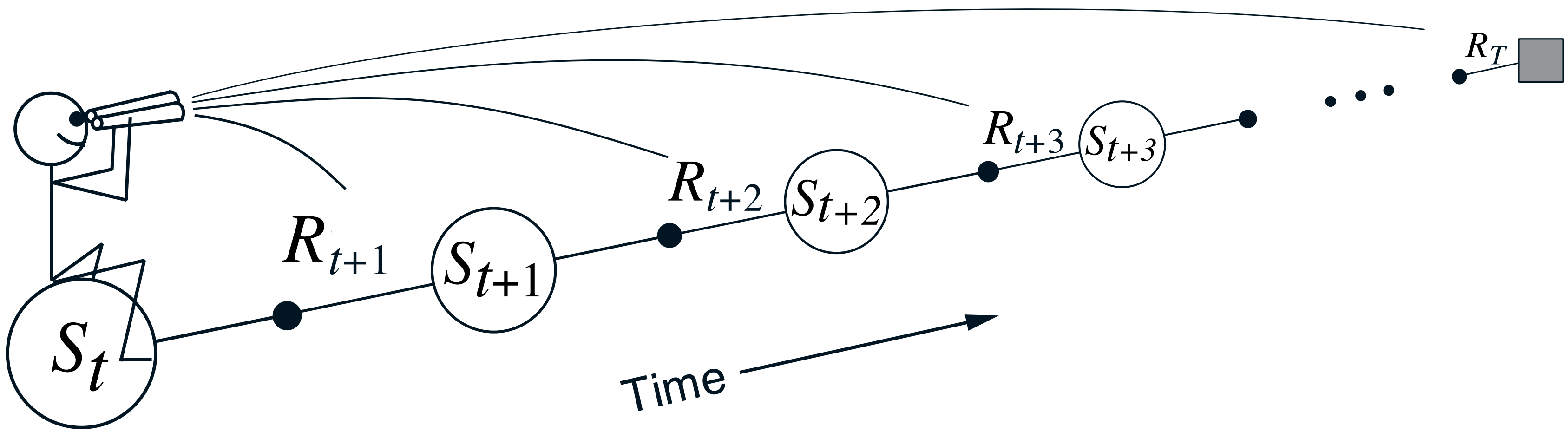
The λ-return alg performs similarly to *n*-step algs on the 19-state random walk (Tabular)



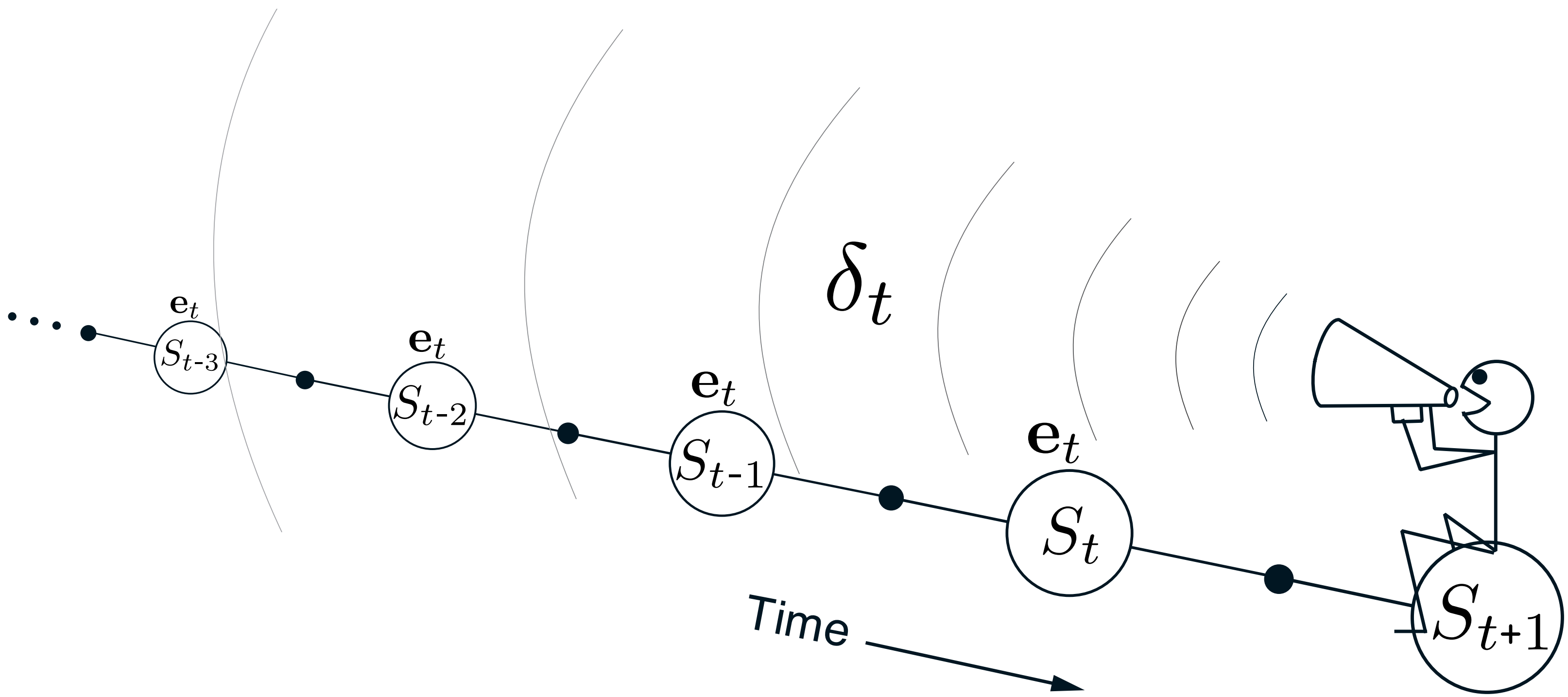
Intermediate λ is best (just like intermediate *n* is best)

λ-return slightly better than *n*-step

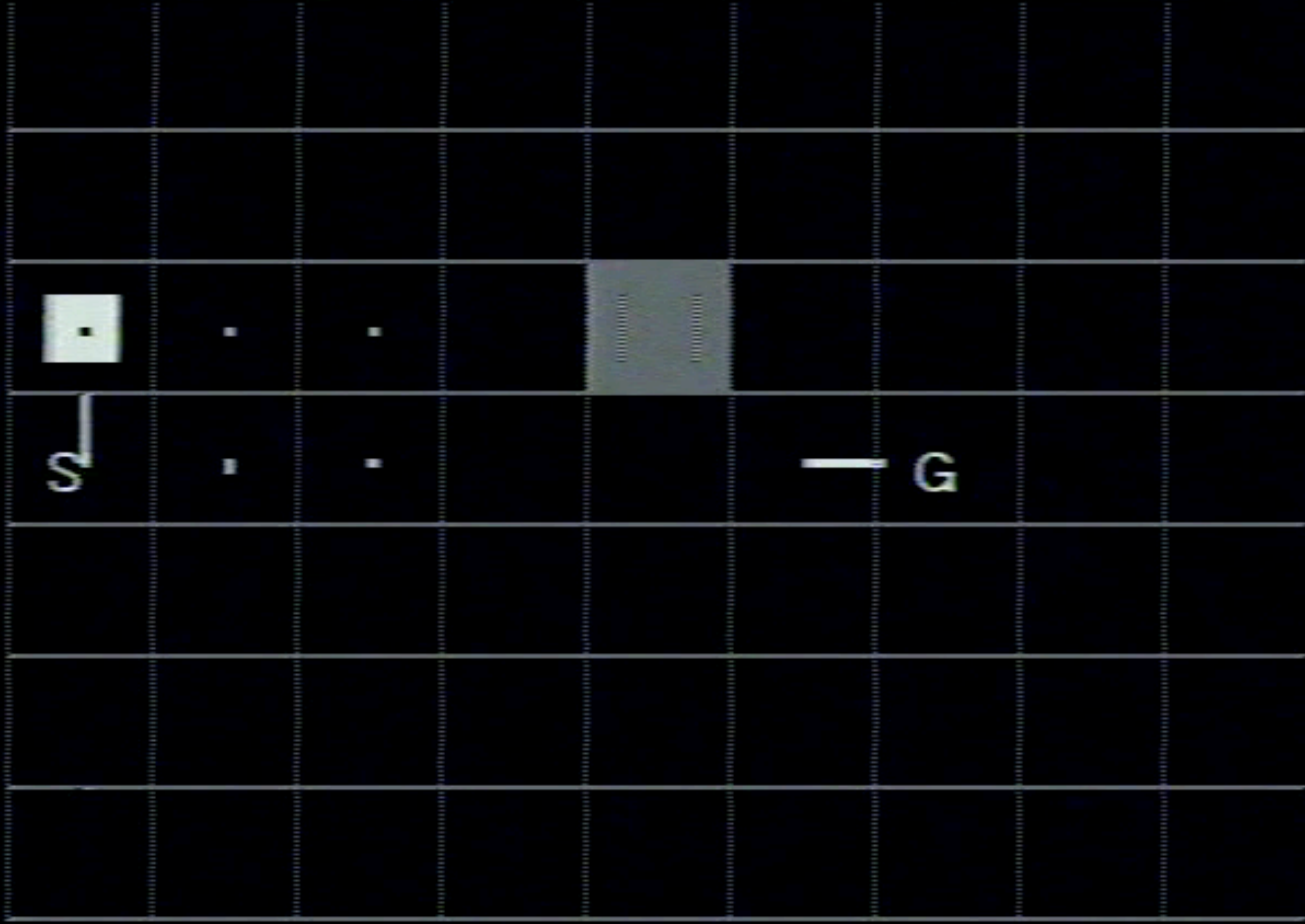
The forward view looks forward from the state being updated to future states and rewards



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 $\gamma\lambda$



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

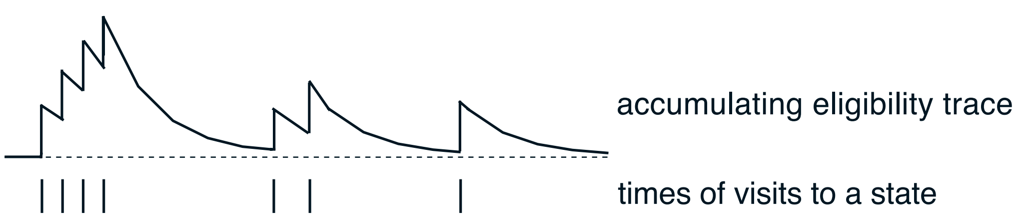
Eligibility traces (mechanism)

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

same shape as $\boldsymbol{\theta}$

- New memory vector called *eligibility trace* $\mathbf{e}_t \in \mathbb{R}^n \geq \mathbf{0}$
  - On each step, decay each component by γλ and increment the trace for the current state by 1
  - *Accumulating trace*

$$\mathbf{e}_0 \doteq \mathbf{0},$$
$$\mathbf{e}_t \doteq \nabla \hat{v}(S_t,\boldsymbol{\theta}_t) + \gamma\lambda \mathbf{e}_{t-1}$$



The Semi-gradient TD(λ) algorithm

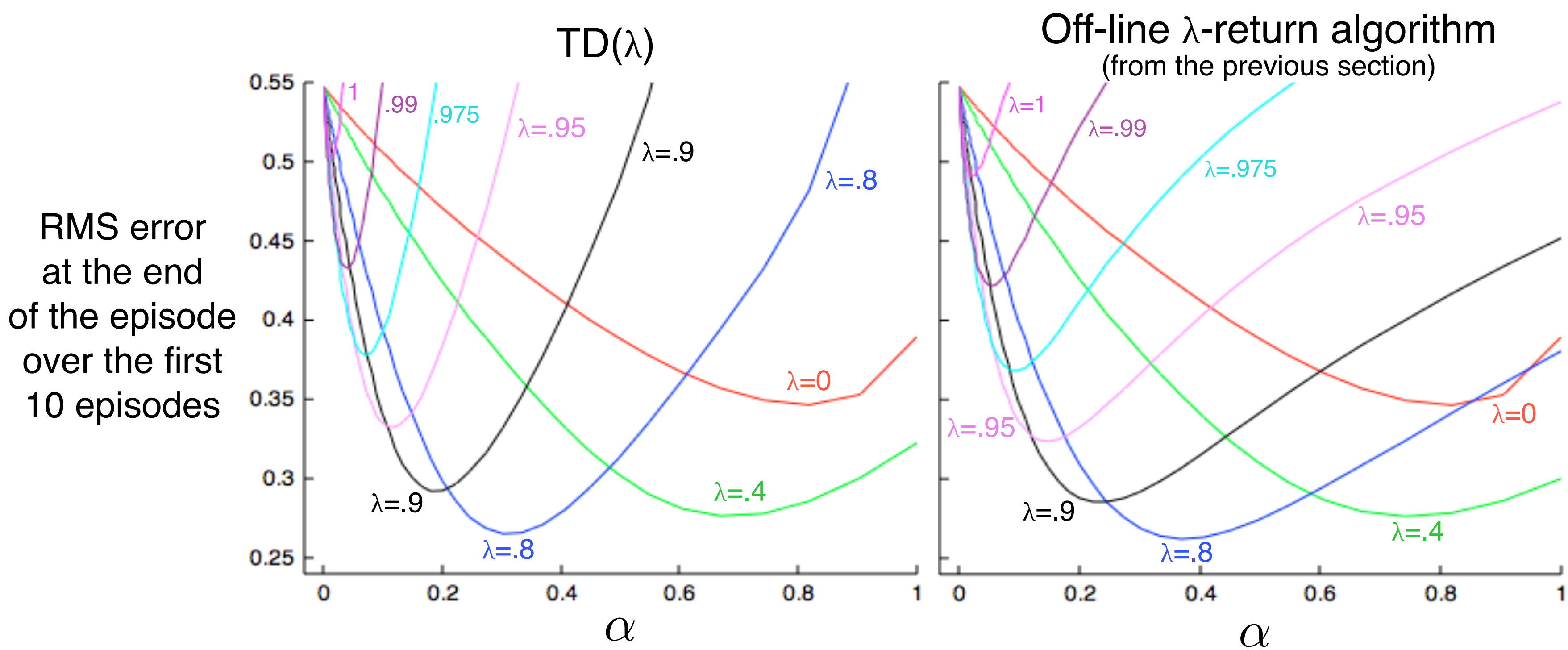
$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \delta_t \mathbf{e}_t$$

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

$$\mathbf{e}_0 \doteq \mathbf{0},$$
$$\mathbf{e}_t \doteq \nabla \hat{v}(S_t, \boldsymbol{\theta}_t) + \gamma \lambda \mathbf{e}_{t-1}$$

TD(λ) performs similarly to offline λ-return alg. but slightly worse, particularly at high α

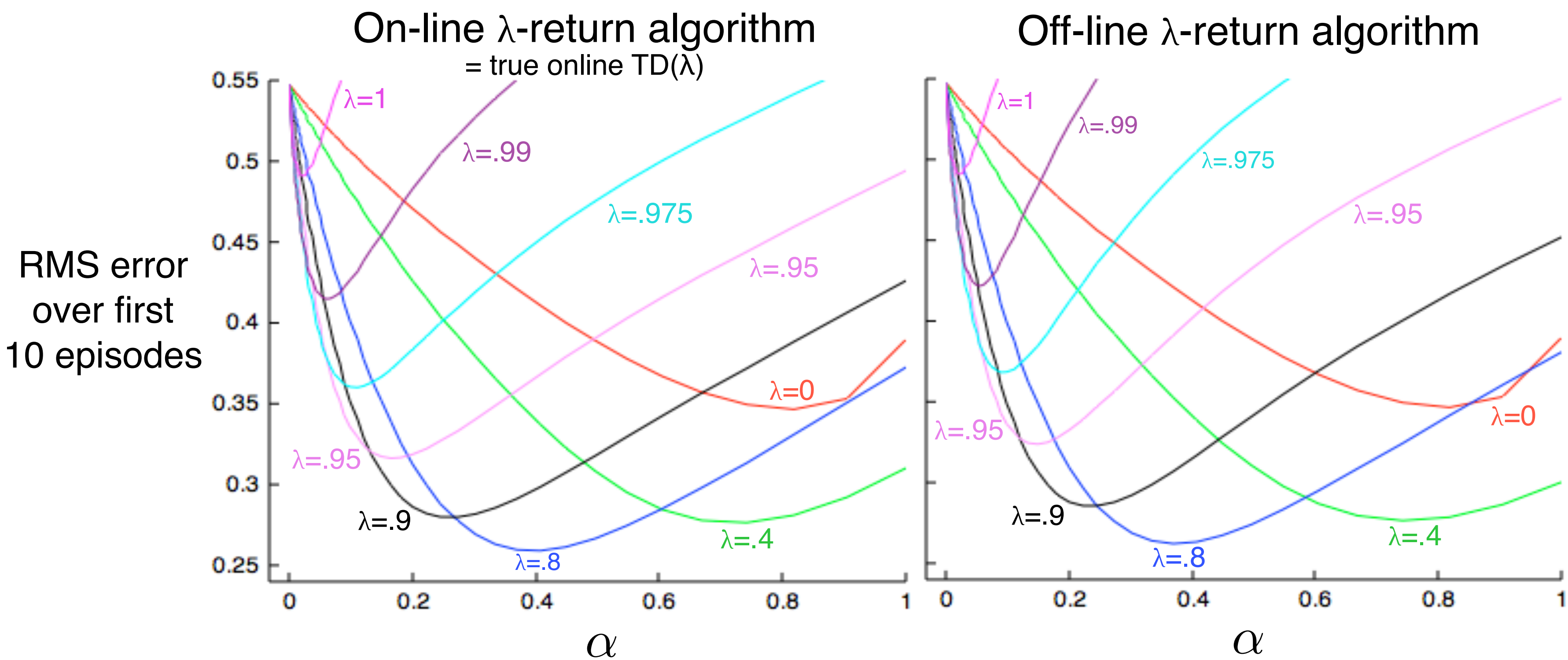
Tabular 19-state random walk task



Can we do better? Can we update online?

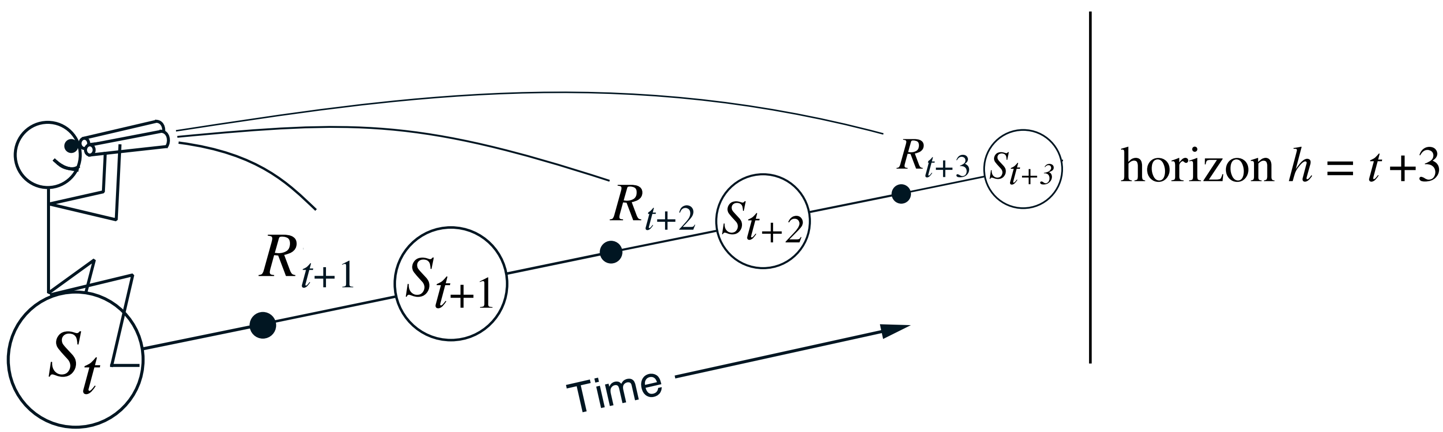
The online λ-return algorithm performs best of all

Tabular 19-state random walk task



The online λ-return alg uses a *truncated λ-return* as its target

$$G_t^{\lambda|h} \doteq (1-\lambda)\sum_{n=1}^{h-t-1}\lambda^{n-1}G_t^{(n)} + \lambda^{h-t-1}G_t^{(h-t)}, \qquad 0 \leq t < h \leq T$$



$$\boldsymbol{\theta}_{t+1}^h \doteq \boldsymbol{\theta}_t^h + \alpha\left[G_t^{\lambda|h} - \hat{v}(S_t,\boldsymbol{\theta}_t^h)\right]\nabla\hat{v}(S_t,\boldsymbol{\theta}_t^h)$$

There is a separate $\boldsymbol{\theta}$ sequence for each h!

The online λ-return algorithm

$$\boldsymbol{\theta}_{t+1}^h \doteq \boldsymbol{\theta}_t^h + \alpha \left[G_t^{\lambda|h} - \hat{v}(S_t,\boldsymbol{\theta}_t^h) \right] \nabla \hat{v}(S_t,\boldsymbol{\theta}_t^h)$$

There is a separate $\boldsymbol{\theta}$ sequence for each h!

$$h = 1: \quad \boldsymbol{\theta}_1^1 \doteq \boldsymbol{\theta}_0^1 + \alpha \left[G_0^{\lambda|1} - \hat{v}(S_0,\boldsymbol{\theta}_0^1) \right] \nabla \hat{v}(S_0,\boldsymbol{\theta}_0^1),$$

$$h = 2: \quad \boldsymbol{\theta}_1^2 \doteq \boldsymbol{\theta}_0^2 + \alpha \left[G_0^{\lambda|2} - \hat{v}(S_0,\boldsymbol{\theta}_0^2) \right] \nabla \hat{v}(S_0,\boldsymbol{\theta}_0^2),$$

$$\boldsymbol{\theta}_2^2 \doteq \boldsymbol{\theta}_1^2 + \alpha \left[G_1^{\lambda|2} - \hat{v}(S_1,\boldsymbol{\theta}_1^2) \right] \nabla \hat{v}(S_1,\boldsymbol{\theta}_1^2),$$

$$h = 3: \quad \boldsymbol{\theta}_1^3 \doteq \boldsymbol{\theta}_0^3 + \alpha \left[G_0^{\lambda|3} - \hat{v}(S_0,\boldsymbol{\theta}_0^3) \right] \nabla \hat{v}(S_0,\boldsymbol{\theta}_0^3),$$

$$\boldsymbol{\theta}_2^3 \doteq \boldsymbol{\theta}_1^3 + \alpha \left[G_1^{\lambda|3} - \hat{v}(S_1,\boldsymbol{\theta}_1^3) \right] \nabla \hat{v}(S_1,\boldsymbol{\theta}_1^3),$$

$$\boldsymbol{\theta}_3^3 \doteq \boldsymbol{\theta}_2^3 + \alpha \left[G_2^{\lambda|3} - \hat{v}(S_2,\boldsymbol{\theta}_2^3) \right] \nabla \hat{v}(S_2,\boldsymbol{\theta}_2^3).$$

$$\vdots$$

$$\begin{array}{cccccc}
\boldsymbol{\theta}_0^0 & & & & & \\
\boldsymbol{\theta}_0^1 & \boldsymbol{\theta}_1^1 & & & & \\
\boldsymbol{\theta}_0^2 & \boldsymbol{\theta}_1^2 & \boldsymbol{\theta}_2^2 & & & \\
\boldsymbol{\theta}_0^3 & \boldsymbol{\theta}_1^3 & \boldsymbol{\theta}_2^3 & \boldsymbol{\theta}_3^3 & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
\boldsymbol{\theta}_0^T & \boldsymbol{\theta}_1^T & \boldsymbol{\theta}_2^T & \boldsymbol{\theta}_3^T & \cdots & \boldsymbol{\theta}_T^T
\end{array}$$

True online TD(λ) computes just the diagonal, cheaply (for linear FA)

True online TD(λ)

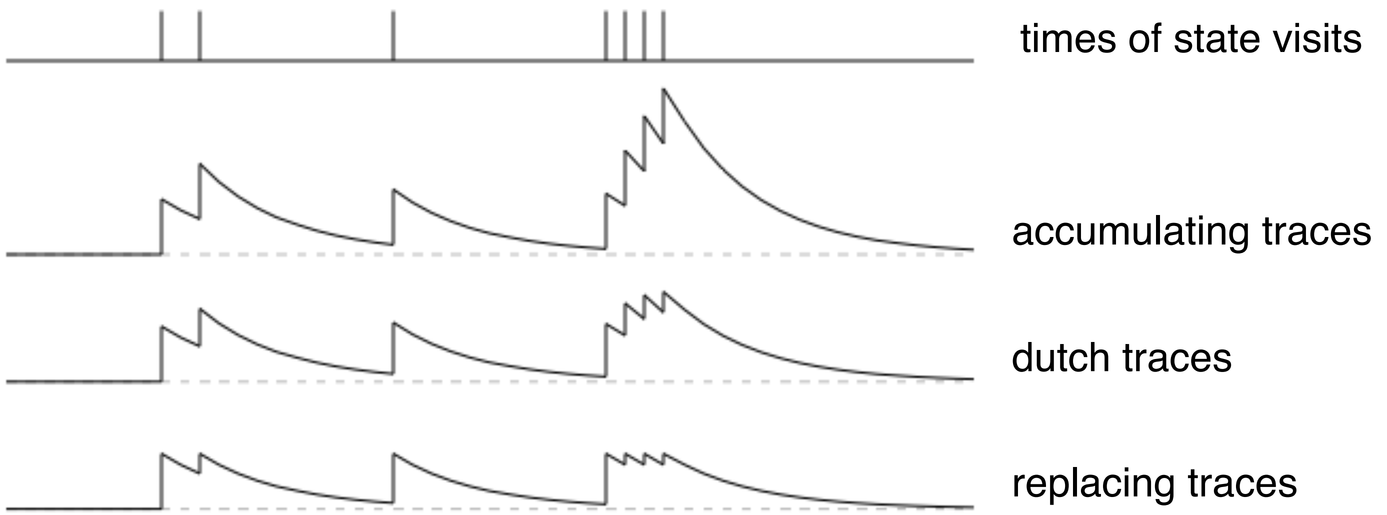
$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \delta_t \mathbf{e}_t + \alpha \left(\boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t - \boldsymbol{\theta}_{t-1}^\top \boldsymbol{\phi}_t \right) (\mathbf{e}_t - \boldsymbol{\phi}_t)$$

$$\mathbf{e}_t \doteq \gamma\lambda \mathbf{e}_{t-1} + \left(1 - \alpha\gamma\lambda \mathbf{e}_{t-1}^\top \boldsymbol{\phi}_t \right) \boldsymbol{\phi}_t$$

dutch trace

Accumulating, Dutch, and Replacing Traces

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



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 Z with linear function approximation

	← episode →						← next episode →		
Time	*0*	*1*	*2*	*. . .*	*T-1*	*T*	*0*	*1*	*2*
Data	ϕ_0	ϕ_1	ϕ_2	$\dots$	ϕ_{T-1}	Z			
Weights	$\boldsymbol{\theta}_0$	$\boldsymbol{\theta}_0$	$\boldsymbol{\theta}_0$	$\dots$	$\boldsymbol{\theta}_0$	$\boldsymbol{\theta}_T$	$\boldsymbol{\theta}_T$	$\boldsymbol{\theta}_T$	$\boldsymbol{\theta}_T$
Predictions $\approx Z$	$\boldsymbol{\theta}_0^\top \phi_0$	$\boldsymbol{\theta}_0^\top \phi_1$	$\boldsymbol{\theta}_0^\top \phi_2$	$\dots$	$\boldsymbol{\theta}_0^\top \phi_{T-1}$				

$$\text{MC:} \quad \boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \left(Z - \phi_t^\top \boldsymbol{\theta}_t\right) \phi_t, \qquad t = 0, \dots, T-1$$

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: $$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \left(Z - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t\right) \boldsymbol{\phi}_t, \qquad t = 0, \ldots, T-1$$

step size

all done at time T

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

MC: $\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \left(Z - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t\right) \boldsymbol{\phi}_t, \qquad t = 0, \ldots, T-1$

step size

all done at time T

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

Final Result

Given:

$$\boldsymbol{\theta}_0 \quad \boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{T-1} \quad Z$$

MC algorithm:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \left(Z - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t\right) \boldsymbol{\phi}_t, \qquad t = 0, \ldots, T-1$$

Equivalent independent-of-span algorithm:

$$\begin{aligned}
&\boldsymbol{\theta}_T \doteq \boldsymbol{a}_{T-1} + Z\boldsymbol{e}_{T-1}, && \boldsymbol{a}_t \in \Re^n, \boldsymbol{e}_t \in \Re^n \\
&\boldsymbol{a}_0 \doteq \boldsymbol{\theta}_0, \text{ then } \boldsymbol{a}_t \doteq \boldsymbol{a}_{t-1} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top \boldsymbol{a}_{t-1}, && t = 1, \ldots, T-1 \\
&\boldsymbol{e}_0 \doteq \alpha_0 \boldsymbol{\phi}_0, \text{ then } \boldsymbol{e}_t \doteq \boldsymbol{e}_{t-1} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t, && t = 1, \ldots, T-1
\end{aligned}$$

Proved:

$$\boldsymbol{\theta}_T = \boldsymbol{\theta}_T$$

MC: $$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \left(Z - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t\right) \boldsymbol{\phi}_t, \qquad t = 0, \dots, T-1$$

$$\begin{aligned}
\boldsymbol{\theta}_T &= \boldsymbol{\theta}_{T-1} + \alpha_{T-1} \left(Z - \boldsymbol{\phi}_{T-1}^\top \boldsymbol{\theta}_{T-1}\right) \boldsymbol{\phi}_{T-1} \\
&= \boldsymbol{\theta}_{T-1} + \alpha_{T-1} \boldsymbol{\phi}_{T-1} \left(-\boldsymbol{\phi}_{T-1}^\top \boldsymbol{\theta}_{T-1}\right) + \alpha_{T-1} Z \boldsymbol{\phi}_{T-1} \\
&= \left(\mathbf{I} - \alpha_{T-1} \boldsymbol{\phi}_{T-1} \boldsymbol{\phi}_{T-1}^\top\right) \boldsymbol{\theta}_{T-1} + Z \alpha_{T-1} \boldsymbol{\phi}_{T-1} \\
&= \mathbf{F}_{T-1} \boldsymbol{\theta}_{T-1} + Z \alpha_{T-1} \boldsymbol{\phi}_{T-1} \qquad\qquad (\text{where } \mathbf{F}_t \doteq \mathbf{I} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top) \\
&= \mathbf{F}_{T-1} \left(\mathbf{F}_{T-2} \boldsymbol{\theta}_{T-2} + Z \alpha_{T-2} \boldsymbol{\phi}_{T-2}\right) + Z \alpha_{T-1} \boldsymbol{\phi}_{T-1} \\
&= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \boldsymbol{\theta}_{T-2} + Z \left(\mathbf{F}_{T-1} \alpha_{T-2} \boldsymbol{\phi}_{T-2} + \alpha_{T-1} \boldsymbol{\phi}_{T-1}\right) \\
&= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \left(\mathbf{F}_{T-3} \boldsymbol{\theta}_{T-3} + Z \alpha_{T-3} \boldsymbol{\phi}_{T-3}\right) + Z \left(\mathbf{F}_{T-1} \alpha_{T-2} \boldsymbol{\phi}_{T-2} + \alpha_{T-1} \boldsymbol{\phi}_{T-1}\right) \\
&= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{F}_{T-3} \boldsymbol{\theta}_{T-3} + Z \left(\mathbf{F}_{T-1} \mathbf{F}_{T-2} \alpha_{T-3} \boldsymbol{\phi}_{T-3} + \mathbf{F}_{T-1} \alpha_{T-2} \boldsymbol{\phi}_{T-2} + \alpha_{T-1} \boldsymbol{\phi}_{T-1}\right) \\
&\ \vdots \\
&= \underbrace{\mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_0 \boldsymbol{\theta}_0}_{\boldsymbol{a}_{T-1}} \;+\; \underbrace{Z \sum_{k=0}^{T-1} \mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_{k+1} \alpha_k \boldsymbol{\phi}_k}_{\boldsymbol{e}_{T-1}} \\
&= \boldsymbol{a}_{T-1} + Z \boldsymbol{e}_{T-1}
\end{aligned}$$

auxiliary short-term-memory vectors $\boldsymbol{a}_t \in \Re^n, \boldsymbol{e}_t \in \Re^n$

$$= \underbrace{\mathbf{F}_{T-1}\mathbf{F}_{T-2}\cdots\mathbf{F}_0\boldsymbol{\theta}_0}_{\boldsymbol{a}_{T-1}} \;+\; \underbrace{Z\sum_{k=0}^{T-1}\mathbf{F}_{T-1}\mathbf{F}_{T-2}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k}_{\boldsymbol{e}_{T-1}}$$

$$= \boldsymbol{a}_{T-1} + Z\boldsymbol{e}_{T-1}$$

$$\begin{aligned}
\boldsymbol{e}_t &\doteq \sum_{k=0}^{t}\mathbf{F}_t\mathbf{F}_{t-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k, \qquad t = 0,\ldots,T-1 \\
&= \sum_{k=0}^{t-1}\mathbf{F}_t\mathbf{F}_{t-1}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k + \alpha_t\boldsymbol{\phi}_t \\
&= \mathbf{F}_t\sum_{k=0}^{t-1}\mathbf{F}_{t-1}\mathbf{F}_{t-2}\cdots\mathbf{F}_{k+1}\alpha_k\boldsymbol{\phi}_k + \alpha_t\boldsymbol{\phi}_t \\
&= \mathbf{F}_t\boldsymbol{e}_{t-1} + \alpha_t\boldsymbol{\phi}_t, \qquad t = 1,\ldots,T-1 \\
&= \boldsymbol{e}_{t-1} - \alpha_t\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top\boldsymbol{e}_{t-1} + \alpha_t\boldsymbol{\phi}_t, \qquad t = 1,\ldots,T-1
\end{aligned}$$

$$\boldsymbol{a}_t \;\doteq\; \mathbf{F}_t\mathbf{F}_{t-1}\cdots\mathbf{F}_0\boldsymbol{\theta}_0 \;=\; \mathbf{F}_t\boldsymbol{a}_{t-1} \;=\; \boldsymbol{a}_{t-1} - \alpha_t\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top\boldsymbol{a}_{t-1}, \quad t = 1,\ldots,T-1$$

Final Result

Given:

$$\boldsymbol{\theta}_0 \quad \boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{T-1} \quad Z$$

MC:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \left(Z - \boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t\right) \boldsymbol{\phi}_t, \qquad t = 0, \ldots, T-1$$

Equivalent independent-of-span algorithm:

$$\begin{aligned} &\boldsymbol{\theta}_T \doteq \boldsymbol{a}_{T-1} + Z\boldsymbol{e}_{T-1}, && \boldsymbol{a}_t \in \Re^n, \boldsymbol{e}_t \in \Re^n \\ &\boldsymbol{a}_0 \doteq \boldsymbol{\theta}_0, \text{ then } \boldsymbol{a}_t \doteq \boldsymbol{a}_{t-1} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top \boldsymbol{a}_{t-1}, && t = 1, \ldots, T-1 \\ &\boldsymbol{e}_0 \doteq \alpha_0 \boldsymbol{\phi}_0, \text{ then } \boldsymbol{e}_t \doteq \boldsymbol{e}_{t-1} - \alpha_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t, && t = 1, \ldots, T-1 \end{aligned}$$

Proved:

$$\boldsymbol{\theta}_T = \boldsymbol{\theta}_T$$

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*

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 TD(λ) is new and best
  - Is exactly equivalent to online λ-return algorithm
- Three varieties of traces: accumulating, dutch, (replacing)
- Traces to control in on-policy and off-policy forms
- Traces do have a small cost in computation (≈x2)