Reinforcement Learning

Eligibility Traces

Stefano Albrecht, Pavlos Andreadis

25 February 2020

(Based on slides by Richard Sutton)

THE UNIVERSITY of EDINBURGH
informatics

Lecture Outline

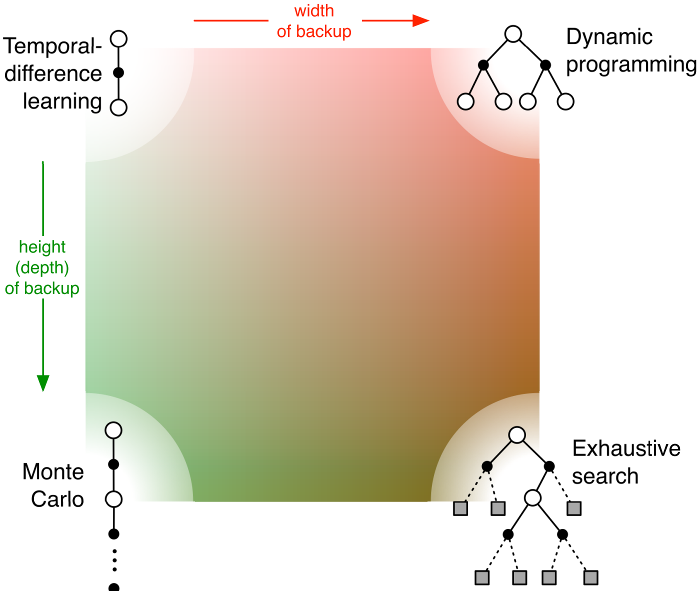
- Compound targets and λ-return
- Offline λ-return algorithm
- Forward view and backward view with eligibility traces
- Semi-gradient TD(λ)
- Online λ-return algorithm and true online TD(λ)
- Approximate control: Sarsa(λ)

Eligibility traces are:

- Another way of interpolating between MC and TD methods
- A basic mechanistic idea — a short-term, fading memory
- Transformation between *forward* and *backward* views
  - Forward view: conceptually simple, good for theory and intuition
  - Backward view: efficient implementation of the forward view

Unified View

n-step TD



Recap: n-Step Returns

n-step returns with function approximation:

- 2-step return:

$$G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2\, \hat{v}(S_{t+2}, \mathbf{w}_{t+1})$$

- n-step return:

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

Use return as update target:

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}\,[\text{Target} - \text{OldEstimate}]$$

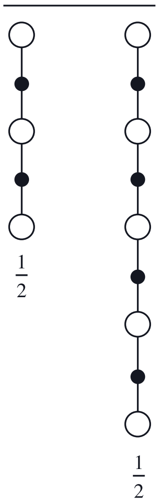
Compound Target

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

- E.g. half a 2-step and half a 4-step

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

A *compound* backup

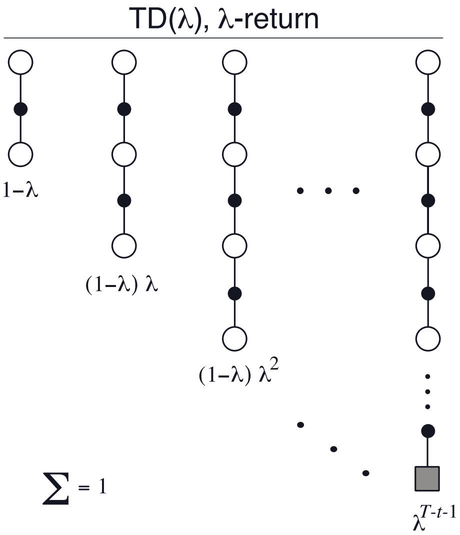


Compound Target: λ-Return

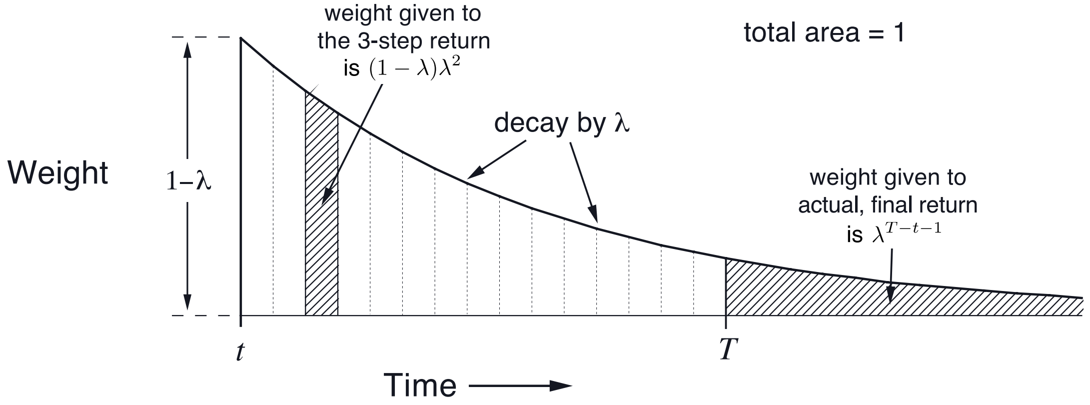
The λ-return is a target that averages *all* n-step returns:

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

- $\lambda \in [0,1]$ is trace-decay parameter



λ-Return Weighting Function



Relation to TD(0) and MC

λ-return can be written as:

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

Relation to TD(0) and MC

λ-return can be written 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 $\lambda = 0$, you get TD(0) target:

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

Relation to TD(0) and MC

λ-return can be written 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 $\lambda = 0$, you get TD(0) target:

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

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

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

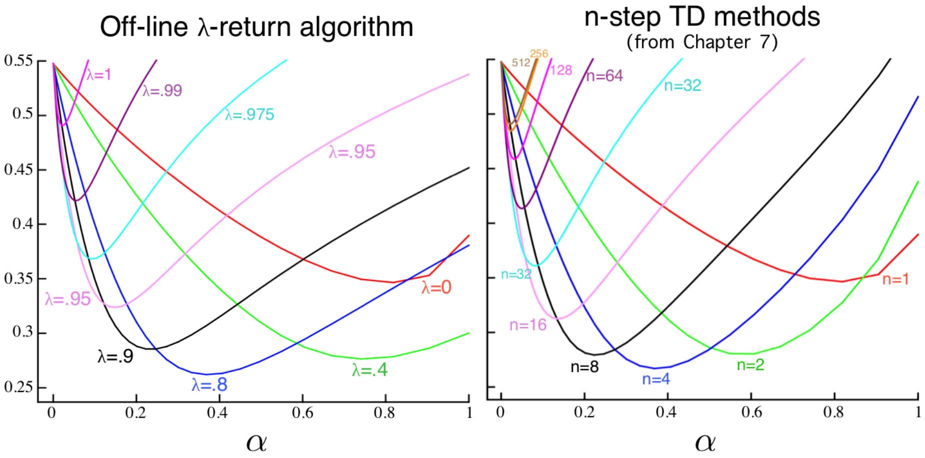
The Offline λ-Return Algorithm

- Wait until the end of the episode
- Then go back over time steps $t = 0, ..., T-1$, updating

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

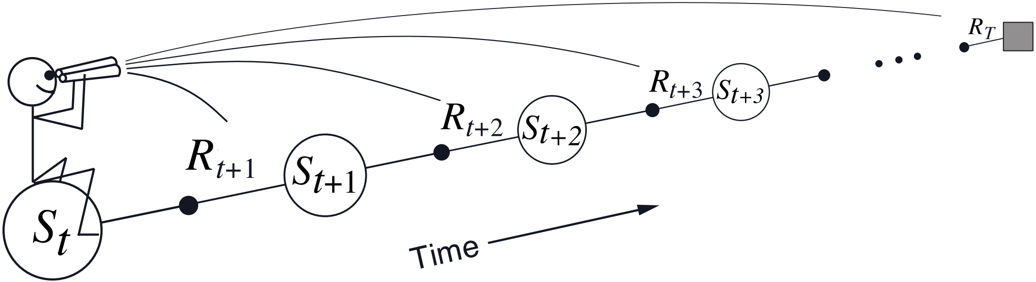
Offline λ-Return vs Tabular n-Step TD in 19-State Random Walk

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



Forward view: looks forward from updated state to future states and rewards

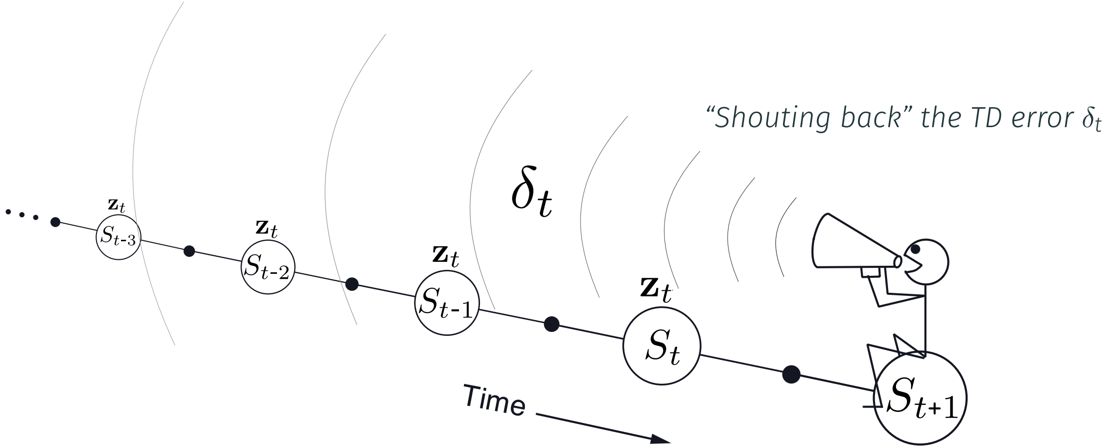
- E.g. MC, n-step TD, offline λ-return algo.



Backward View

Backward view: looks back to recently visited states

- More efficient implementation of forward view using *eligibility traces*



Eligibility Trace

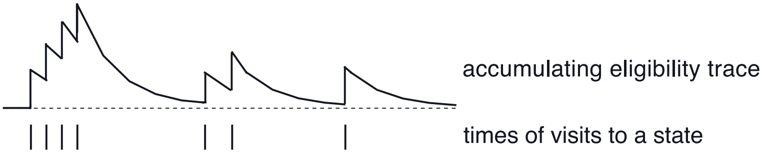
Value function parameters $\mathbf{w}_t \in \mathbb{R}^d$ act as *long-term memory*

Eligibility trace vector $\mathbf{z}_t \in \mathbb{R}^d$ acts as *short-term memory*

- On each step, decay each component of $\mathbf{z}_t$ by $\gamma\lambda$ and increment the trace for current state S_t
- E.g. accumulating tace:

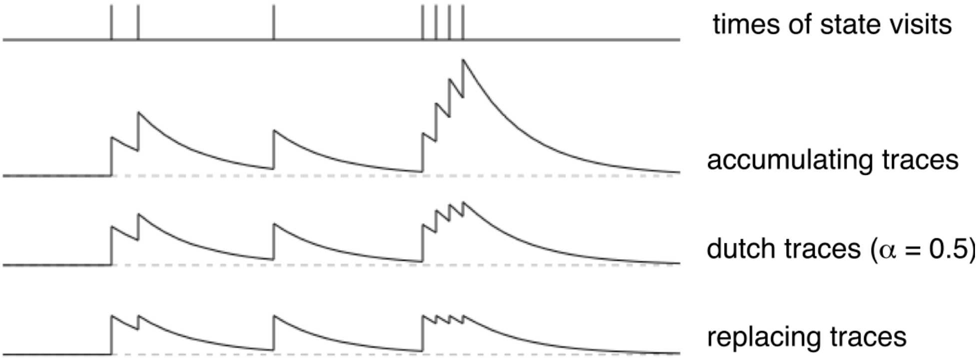
$$\mathbf{z}_{-1} \doteq 0$$

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



Accumulating, Dutch, and Replacing Traces

- All traces fade the same: by $\gamma\lambda$
- But may increment differently:

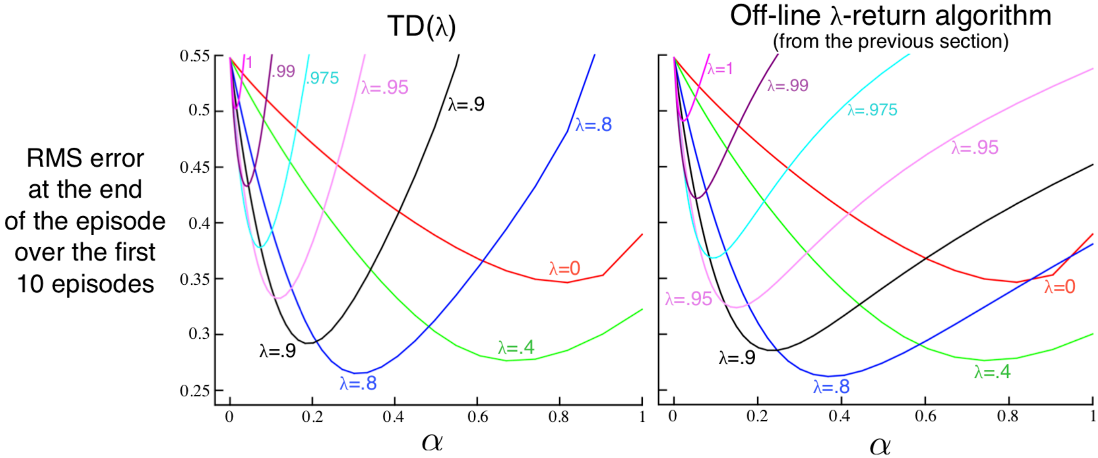


Semi-Gradient TD(λ)

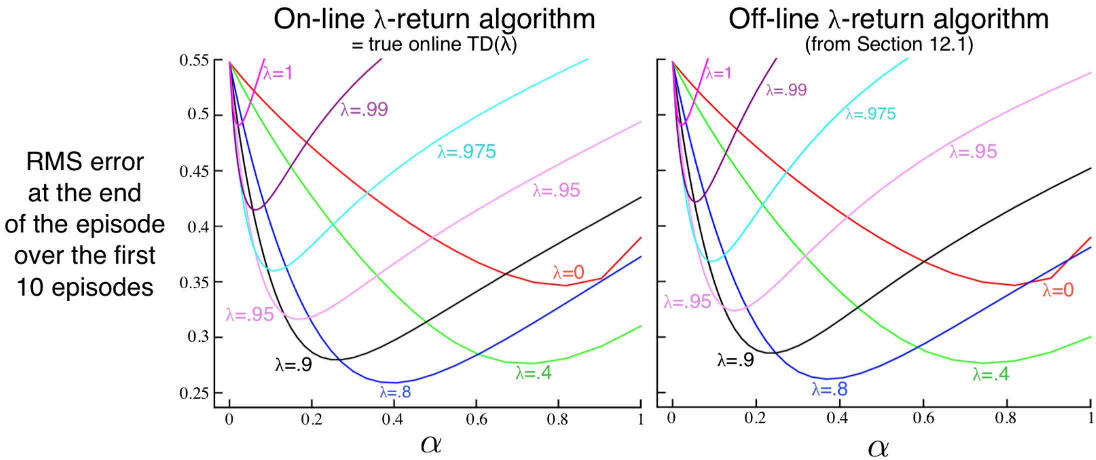
Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  $\mathbf{z} \leftarrow \mathbf{0}$
  Loop 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

Semi-Gradient TD(λ) vs Offline λ-Return in 19-State Random Walk



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



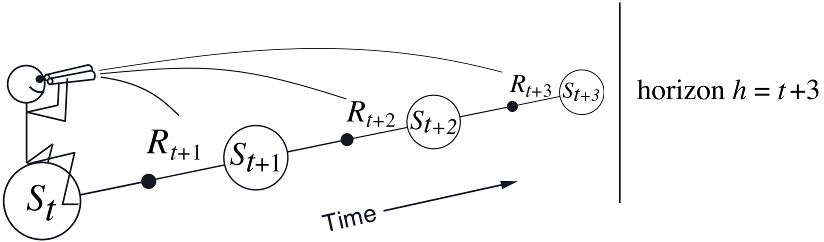
Online λ-return algorithm performs best of all (+ updates online)

Online λ-Return Algorithm

Online λ-return algorithm uses truncated λ-return:

$$G^{\lambda}_{t:h} = (1-\lambda)\sum_{n=1}^{h-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{h-t-1} G_{t:h}, \quad 0 \le t < h \le T$$

- $G_{t:h}$ is longest available n-step return from time t
- In each new time step, “redo” all updates since beginning of episode



Online λ-Return Algorithm — Update Sequence

$$\mathbf{w}_{t+1}^{h} \doteq \mathbf{w}_{t}^{h} + \alpha \left[G_{t:h}^{\lambda} - \hat{v}(S_t, \mathbf{w}_t^h) \right] \nabla \hat{v}(S_t, \mathbf{w}_t^h), \quad 0 \leq t < h \leq T$$

$$h = 1: \quad \mathbf{w}_1^1 \doteq \mathbf{w}_0^1 + \alpha \left[G_{0:1}^{\lambda} - \hat{v}(S_0, \mathbf{w}_0^1) \right] \nabla \hat{v}(S_0, \mathbf{w}_0^1),$$

$$\begin{aligned} h = 2: \quad \mathbf{w}_1^2 &\doteq \mathbf{w}_0^2 + \alpha \left[G_{0:2}^{\lambda} - \hat{v}(S_0, \mathbf{w}_0^2) \right] \nabla \hat{v}(S_0, \mathbf{w}_0^2), \\ \mathbf{w}_2^2 &\doteq \mathbf{w}_1^2 + \alpha \left[G_{1:2}^{\lambda} - \hat{v}(S_1, \mathbf{w}_1^2) \right] \nabla \hat{v}(S_1, \mathbf{w}_1^2), \end{aligned}$$

$$\begin{aligned} h = 3: \quad \mathbf{w}_1^3 &\doteq \mathbf{w}_0^3 + \alpha \left[G_{0:3}^{\lambda} - \hat{v}(S_0, \mathbf{w}_0^3) \right] \nabla \hat{v}(S_0, \mathbf{w}_0^3), \\ \mathbf{w}_2^3 &\doteq \mathbf{w}_1^3 + \alpha \left[G_{1:3}^{\lambda} - \hat{v}(S_1, \mathbf{w}_1^3) \right] \nabla \hat{v}(S_1, \mathbf{w}_1^3), \\ \mathbf{w}_3^3 &\doteq \mathbf{w}_2^3 + \alpha \left[G_{2:3}^{\lambda} - \hat{v}(S_2, \mathbf{w}_2^3) \right] \nabla \hat{v}(S_2, \mathbf{w}_2^3). \\ &\vdots \end{aligned}$$

True Online TD(λ)

True online TD(λ) computes just the sequence $\mathbf{w}_0^0, \mathbf{w}_1^1, \ldots, \mathbf{w}_T^T$ (for linear FA)

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t + \alpha\left(\mathbf{w}_t^\top\mathbf{x}_t - \mathbf{w}_{t-1}^\top\mathbf{x}_t\right)(\mathbf{z}_t - \mathbf{x}_t)$$

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

where $\mathbf{x}_t \doteq \mathbf{x}(S_t)$

Produces identical parameter updates to online λ-return algorithm

$\Rightarrow$ But less expensive: memory and time complexity same as TD(λ) — $O(d)$

Control With Eligibility Traces

Control done by usual extension to action-values:

- Forward view:

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \, \hat{q}(S_{t+n}, A_{t+n} \mathbf{w}_{t+n-1}), \quad t+n < T$$

$$G_{t:t+n} \doteq G_t, \quad t+n \geq T$$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[G^{\lambda}_{t:\infty} - \hat{q}(S_t, A_t, \mathbf{w}_t) \right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \quad t = 0, ..., T-1$$

- Backward approximation with Sarsa(λ):

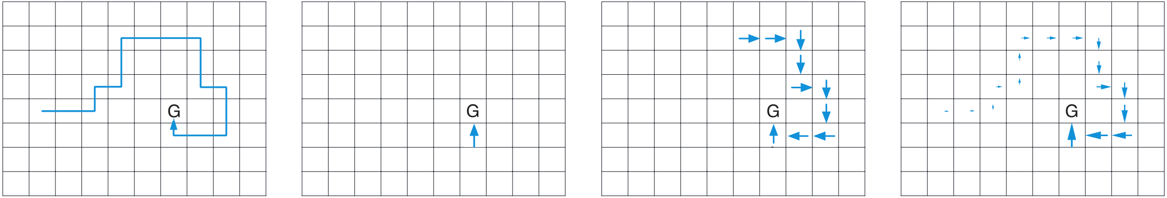
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \, \delta_t \, \mathbf{z}_t$$

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

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

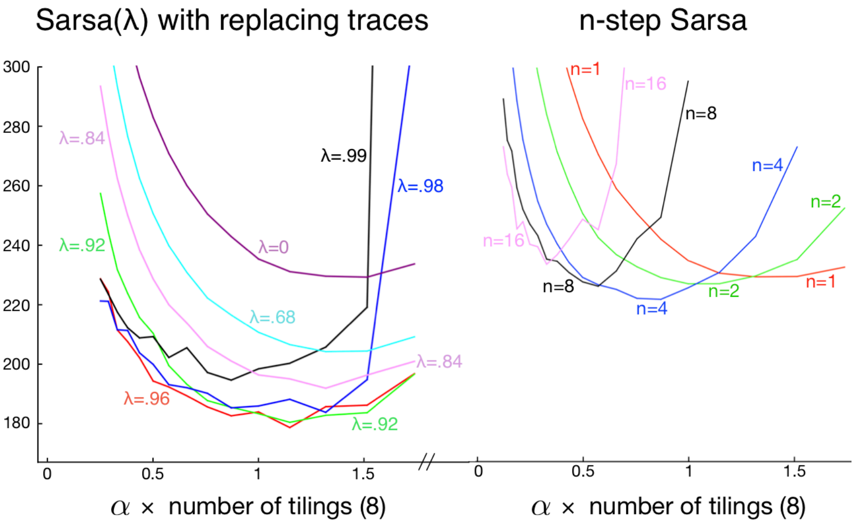
Sarsa(λ) vs n-Step Sarsa in Grid World Example

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



Sarsa(λ) vs n-Step Sarsa in Mountain Car Example

Mountain Car
Steps per episode
averaged over
first 50 episodes
and 100 runs



Reading

Required:

- RL book, chapter 12 (12.1–12.5, 12.7)