Chapter 12
Eligibility Traces

Eligibility Traces (적격성 추적)

Q-Learning, Sarsa

- Eligibility Trace 와 TD를 함께 사용하면 학습을 더 효과적으로 수행할 수 있는 일반식을 만들 수 있음.

General Method

- Eligibility Trace 로 TD, MC 를 통합하여 표현할 수 있음.
  - λ=1 : Monte Carlo (method at one end)
  - λ=0 : one-step TD

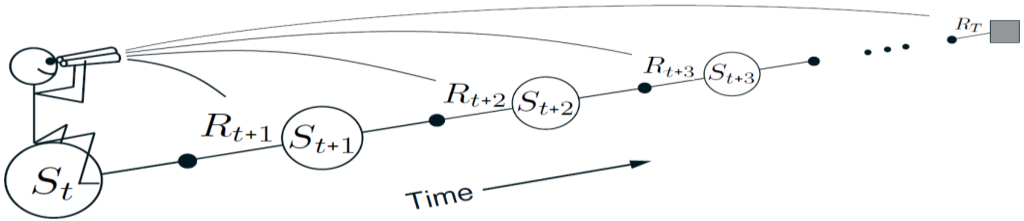
Eligibility Traces 의 특징

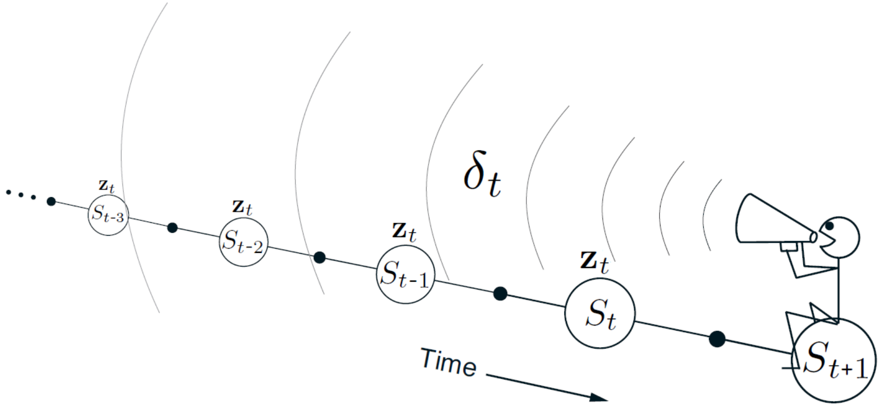
- Eligibility Traces 의 특징
  - Short-term memory vector, the eligibility traces $\mathbf{z}_t \in \mathbb{R}^d$
  - Long-term weight vector $\mathbf{w}_t \in \mathbb{R}^d$
- Advantages
  - Only a single Trace Vector is required (자원)
    - Vs n feature vectors
  - Learning occurs continually and uniformly in time (시간)
    - Vs delayed or end of the episode
  - Learning can occur and affect behavior immediately (효율)
    - vs being delayed n steps

Forward views and Backward views

- Forward views
  - 앞의 보상을 모르고 update 하는 공식들.
    - 미래 보상(현재 유효하지 않음) 보기 때문에 구현 어려움.
- Backward views
  - 현재 TD Error 와 과거(backward) 방문 state 의 eligibility trace 사용

Forward views and Backward views





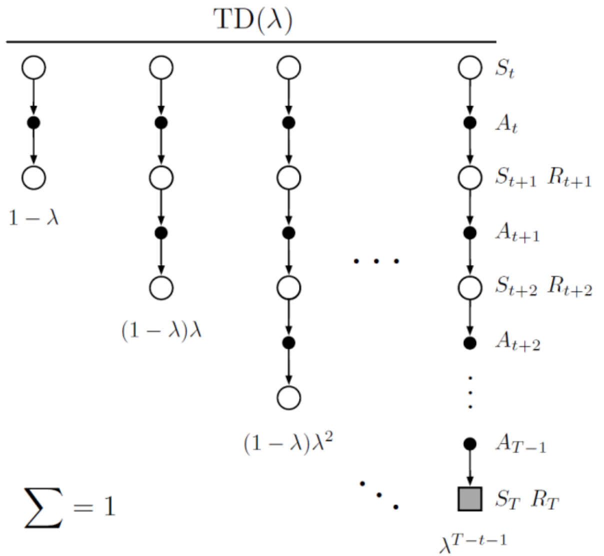
MC, TD and n-step TD

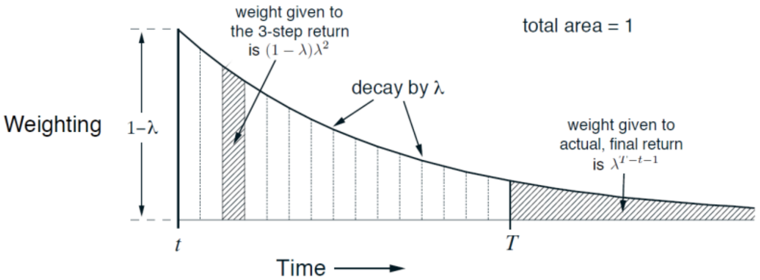
- MC : $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$
- TD : $G_t^{(1)} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
- 2 step TD : $G_t^{(2)} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_t(S_{t+2})$
- N step TD: $G_t^{(n)} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_t(S_{t+n})$ (7.1)
- N step TD with state and weight vector (12.1)

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1}), \quad 0 \leq t \leq T-n,$$

Weighting and decay

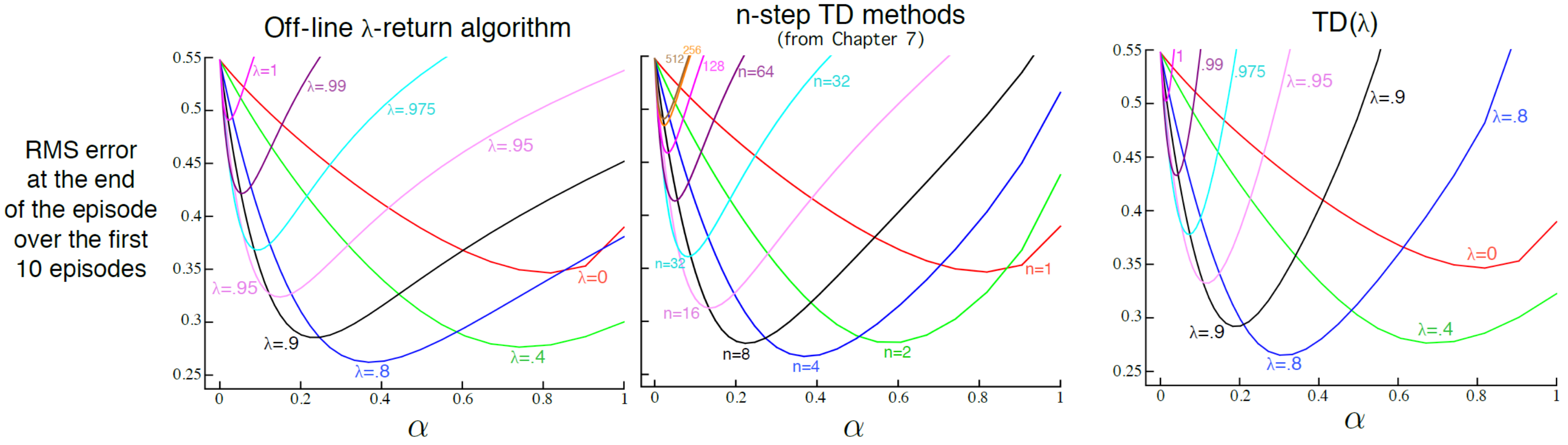
- TD(λ) 의 weighting :
  - 처음 것은 1- λ
  - 이후 weight 합이 1이 되도록 λ만큼 계속 곱하게 됨.
- Decay
  - 갈 수록 weight 는 0 수렴
- If..
  - λ = 1 : MC
  - λ = 0 : one-step TD





RMS comparison

- Off-line λ-return algorithm is slight better at best a and λ, and at high a
- TD(λ) is worse at high a value



SARSA(λ)

- From ch. 10,

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

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

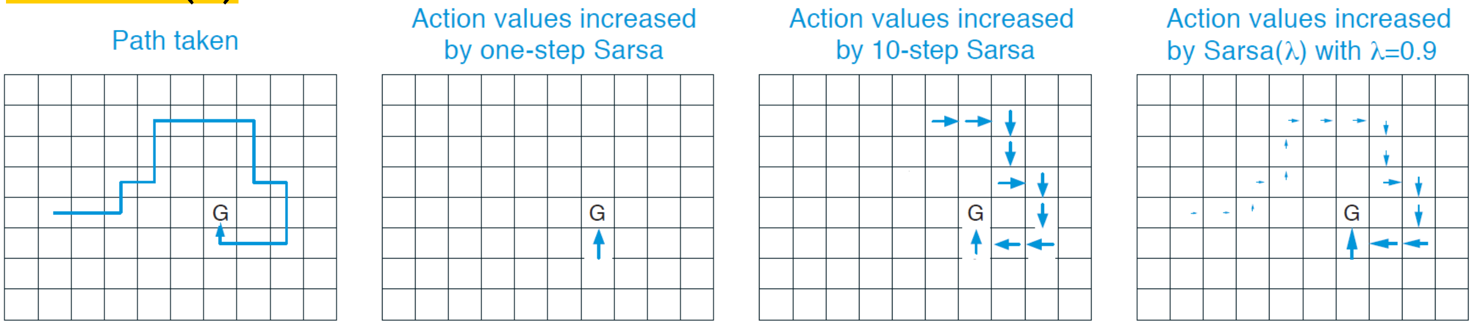
- Action value, TD error and eligibility trace for Sarsa(λ)

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

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

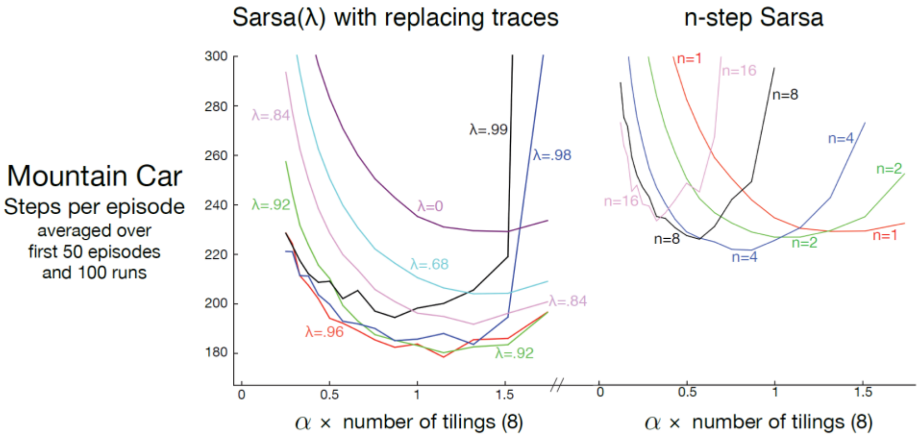
$$\mathbf{z}_{-1} \doteq \mathbf{0},$$
$$\mathbf{z}_t \doteq \gamma \lambda \mathbf{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \quad 0 \le t \le T.$$

SARSA(λ)

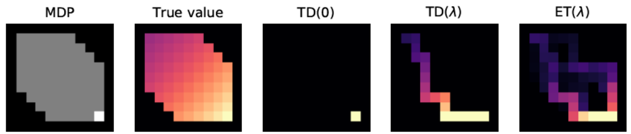


- 1. trace in grid world

- 2. Sarsa(λ) @ ex 10.1
  - At Sarsa(λ), more efficient learning.

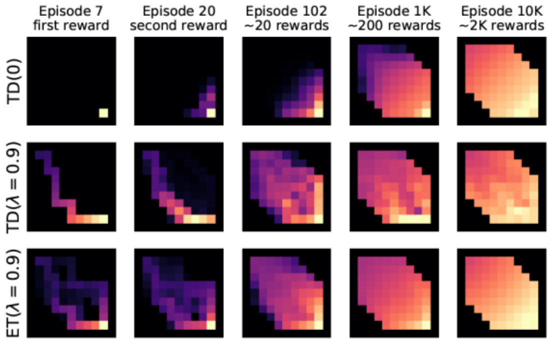


Performance



- TD(0) only updated the last state
- TD(λ) updated the trajectory in this episode
- ET(λ) additionally updated trajectories from earlier (unrewarding) episodes.

- TD(0) learns slowly but steadily,
- TD(λ) learns faster but with higher variance
- ET(λ) learns both fast and stable.



https://arxiv.org/pdf/2007.01839.pdf

Performance

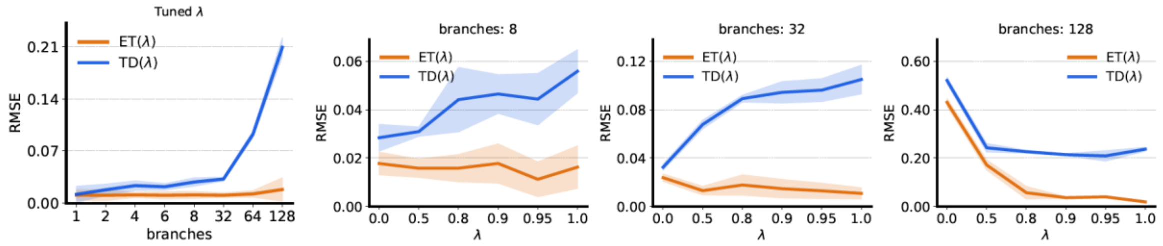


Figure 4: Prediction errors in the multi-chain. ET(λ) (**orange**) consistently outperformed TD(λ) (**blue**). Shaded areas depict standard errors across 10 seeds.

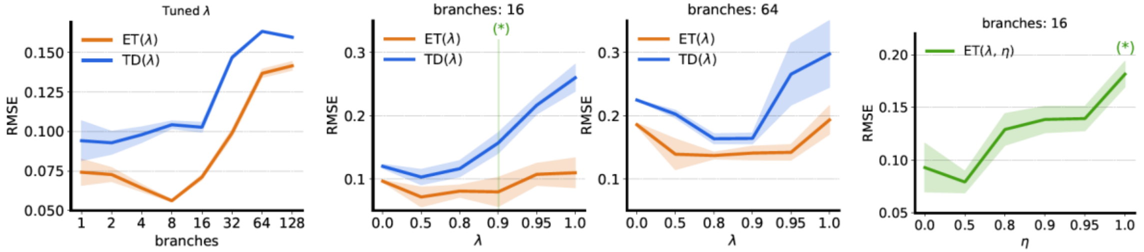


Figure 5: Comparing value error with linear function approximation a) as function of the number of branches (left), b) as function of λ (center two plots) and c) as function of η (right). The left three plots show comparisons of TD(λ) (**blue**) and ET(λ) (**orange**), showing ET(λ) attained lower prediction errors. The right plot interpolates between these algorithms via ET(λ, η), from ET(λ) = ET(λ, 0) to ET(λ, 1) = TD(λ), with $\lambda = 0.9$ (corresponding to a vertical slice indicated in the second plot).

https://arxiv.org/pdf/2007.01839.pdf

Thanks!

Any questions ?