MDP를 모른다니 일단 해보자

Lecture 4: Model-Free Prediction

모델을 모름.

Value를 찾는것.

David Silver

1강 : 강화학습의 개념

2강 : MDP에 대한 설명

3강 : MDP를 알 때 (model을 알 때) 문제푸는 법에 대해 설명

Prediction : Value를 찾는것. (Policy가 정해져 있어야함)

Control : 더 나은 Policy를 찾는것.

Outline

environment를 모를 때, Model-free Prediction을 풀기 위해 ① MC 를 사용한다.
② TD

Prediction은 Policy가 정해져있고 그 Policy를 따를 때의 value fn을 구하는 것. → true value를 아는 것이다.

"optimal value fn 인지는 알 수 없다"

↳ 그래서 Prediction 에서는 Bellman expectation Eq 사용.

Control은 Bellman optimality Eq 사용.

Model-Free Reinforcement Learning

- Last lecture:
  - Planning by dynamic programming
  - Solve a *known* MDP
- This lecture:
  - Model-free prediction
  - Estimate the value function of an *unknown* MDP
- Next lecture:
  - Model-free control
  - Optimise the value function of an *unknown* MDP

Monte-Carlo Reinforcement Learning

↳ empirical, 경험에 의해 구하는 방법
실제로 해보면서 값을 추정하는 것이다.

MDP transition / reward 를 라도 되기 때문에 model-free
↳ episode 해보면서 학습하기 때문에

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from *complete* episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to *episodic* MDPs
  - All episodes must terminate

모든 episode는 끝날수 있어야한다.

MC는 episode가 끝나야 Return (누적된 reward들)을 받아서 학습할 수 있다.
이 return을 평균을 낸것을 value라고 하는 것이 MC.

Monte-Carlo Policy Evaluation = Prediction

- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, ..., S_k \sim \pi$$

- Recall that the *return* is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$$

- Recall that the value function is the expected return:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

확률변수임. 매 episode마다 달라지니까

- Monte-Carlo policy evaluation uses *empirical mean* return instead of *expected* return

First-Visit Monte-Carlo Policy Evaluation

처음 방문한 State만 고려한다.

$$V(s) = \frac{\text{return한 } S(s)}{\text{방문횟수 } N(s)}$$

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$ //first visit은 한번만, every visit은 전부다 증가
- Increment total return $S(s) \leftarrow S(s) + G_t$ total return ↑ → 방문횟수
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

큰수의 법칙

agent가 대체적으로 모든 state를 다 방문해야함
↳ 그래야 $N(s) \rightarrow \infty$ 가 되니까.

표본 수가 많을수록 실제의 확률이 올바른 예측에서 오차가 줄어든다.

Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- Every time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

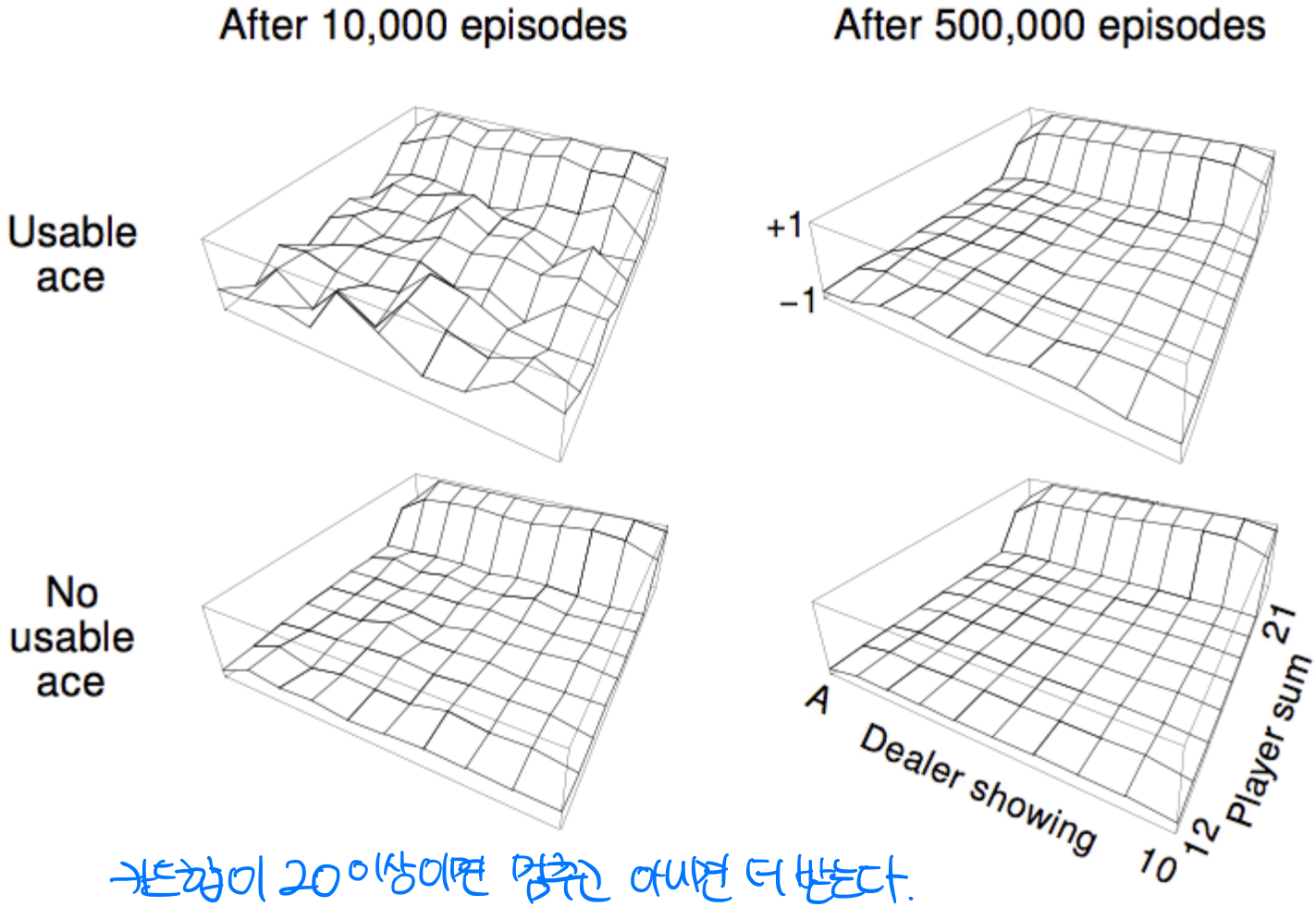
first visit 같음.

Blackjack Example

rule: 그림을 받고 카드를 더 받거나 안받을 수 있다.
최대한 21에 가깝게 하는 것이 goal

- States (200 of them):
  - Current sum (12-21) → 현재 카드의 합
  - Dealer's showing card (ace-10) → 딜러가 보여준 카드
  - Do I have a "useable" ace? (yes-no) → ace 있는가
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - $+1$ if sum of cards $>$ sum of dealer cards
  - 0 if sum of cards $=$ sum of dealer cards
  - -1 if sum of cards $<$ sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12

Blackjack Value Function after Monte-Carlo Learning



카드합이 20이상이면 멈추는 게임이 더 많다.

Policy: stick if sum of cards ≥ 20, otherwise twist

Incremental Mean

The mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally,

MC에서 평균을 구하는 것은 매 episode 마다 방문한 state수 만큼 값을 가지고서 평균을 내야함.
그러나 Incremental Mean을 사용하면 다 저장하지 않아도 된다.

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}$$

$\mu_{k-1} = \frac{1}{k-1}\sum_{j=1}^{k-1} x_j$

$= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right)$

계산해보면 됨.

Incremental Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, ..., S_T$
- For each state S_t with return G_t

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

error term

$N(S_t) \to \infty$ ⇒ 점점 작아진다

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

MDP가 계속 바뀌는 경우.

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

learning rate

고정시킬 수 있다.

⇒ 예전에 한 경험을 잊음.

Temporal-Difference Learning

Model free

TD는 에피소드 끝나지 않아도 사용가능함

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

MC ⇒ G_t 쪽으로 update
TD ⇒ $R_{t+1} + \gamma V(S_{t+1})$ 쪽으로 update

MC and TD

- Goal: learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
  - Update value $V(S_t)$ toward *actual* return G_t

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

TD error
TD target
한 step 가서 예측한 값으로 V(St) 업데이트한다.

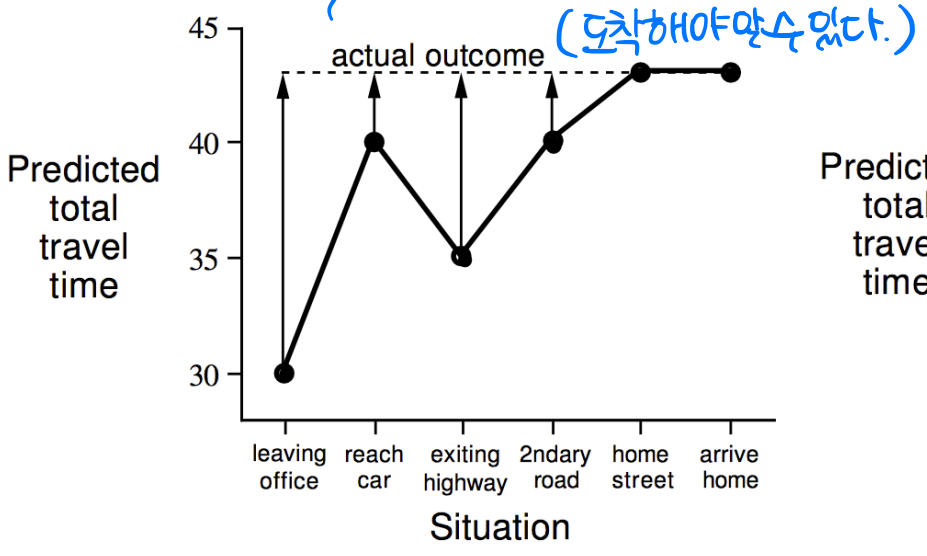
  - $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

Driving Home Example

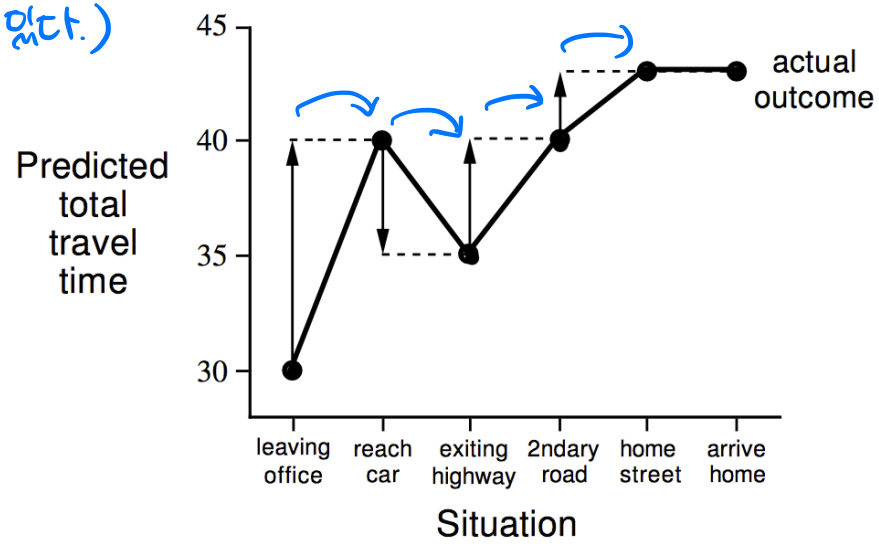
State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

Driving Home Example: MC vs. TD

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



Advantages and Disadvantages of MC vs. TD

MC는 에피소드가 다 끝나야 return을 알 수 있다. (terminate)
TD는 매 step 후 학습하기 때문에 에피소드 다 끝나지 않아도 된다. (non-terminate)

- TD can learn *before* knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn *without* the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

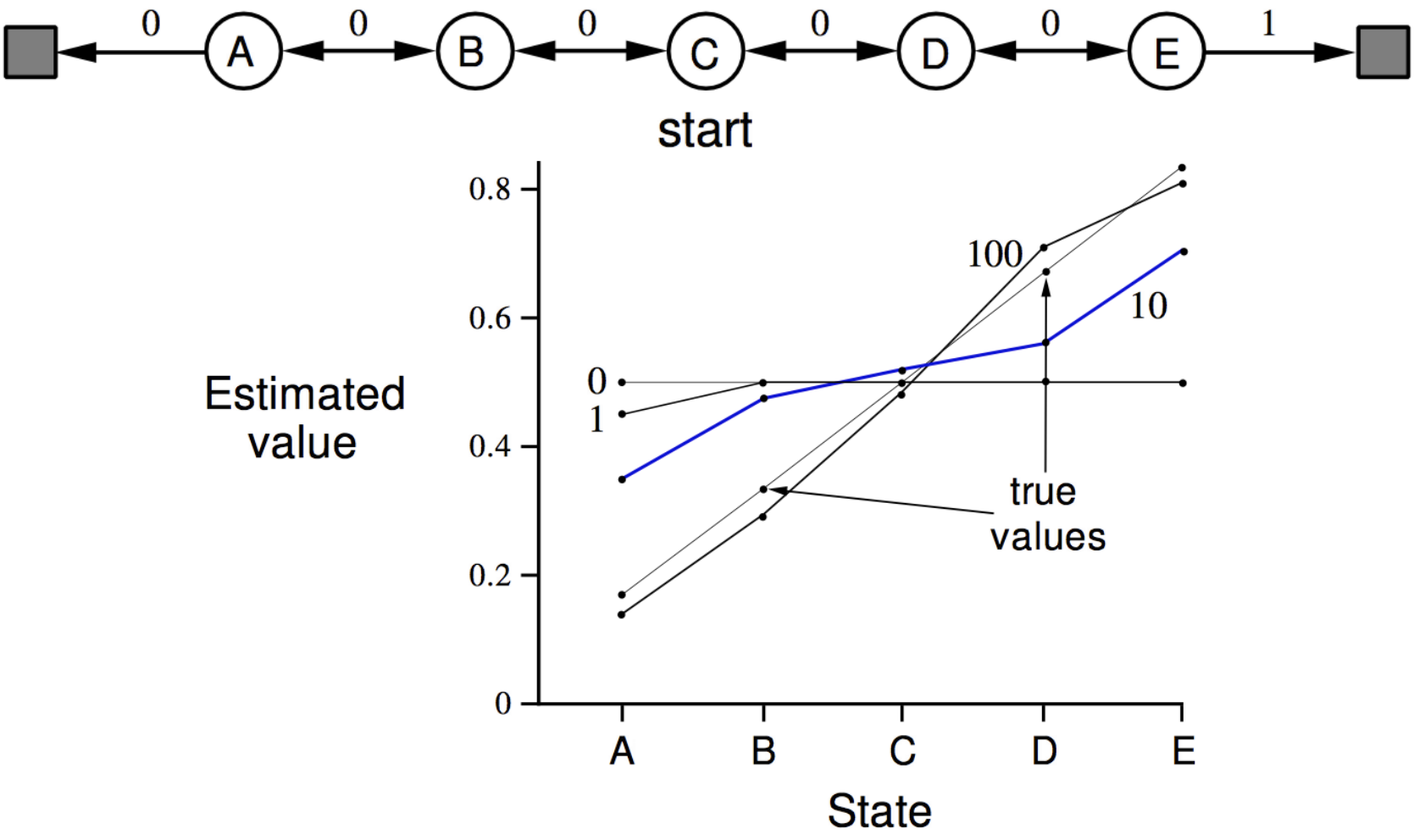
Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:
  - Return depends on *many* random actions, transitions, rewards
  - TD target depends on *one* random action, transition, reward

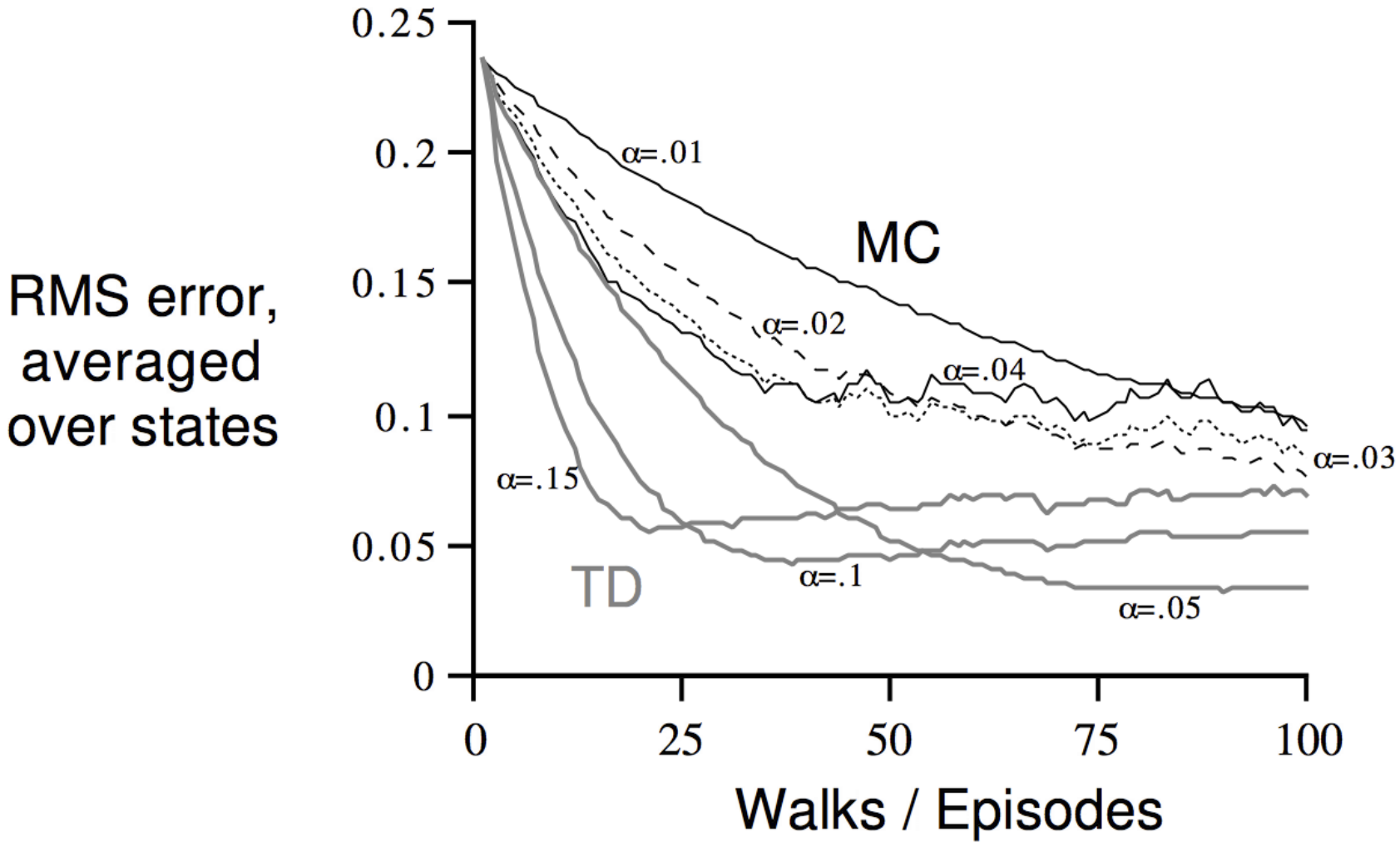
Advantages and Disadvantages of MC vs. TD (2)

- MC has high variance, zero bias
  - Good convergence properties 수렴이 좋다.
  - (even with function approximation) → NN
  - Not very sensitive to initial value 초기값에 덜 민감.
  - Very simple to understand and use
- TD has low variance, some bias
  - Usually more efficient than MC
  - TD(0) converges to $v_\pi(s)$
  - (but not always with function approximation) 수렴 보장이 없다.
  - More sensitive to initial value 초기값에 민감.

Random Walk Example



Random Walk: MC vs. TD



Batch MC and TD → 제한된 데이터가 있는 경우.

- MC and TD converge: $V(s) \to v_\pi(s)$ as experience $\to \infty$
- But what about batch solution for finite experience?

$$s_1^1, a_1^1, r_2^1, ..., s_{T_1}^1$$
$$\vdots$$
$$s_1^K, a_1^K, r_2^K, ..., s_{T_K}^K$$

  - e.g. Repeatedly sample episode $k \in [1, K]$
  - Apply MC or TD(0) to episode k

AB Example

Two states A, B; no discounting; 8 episodes of experience

$$\mathrm{A}, 0, \mathrm{B}, 0$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 1$$
$$\mathrm{B}, 0$$

What is $V(A), V(B)$?

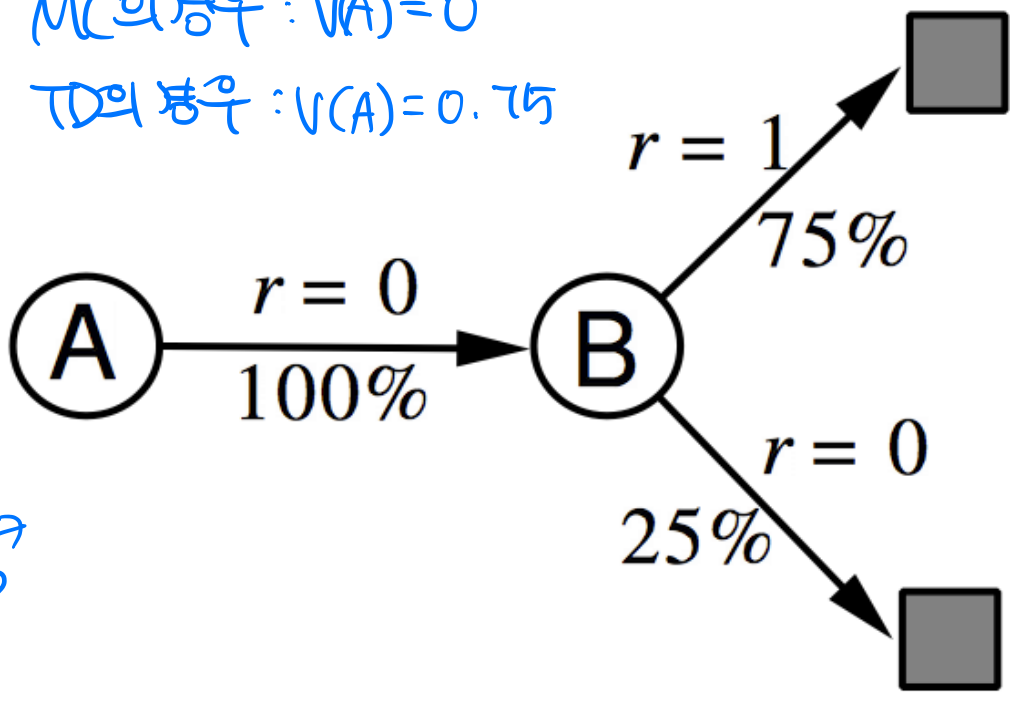
AB Example

Two states A, B; no discounting; 8 episodes of experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0

MC의 경우 : V(A) = 0
TD의 경우 : V(A) = 0.75

8번 중 6번 B 1
V(B) = 0.75



전체 MDP

What is $V(A), V(B)$?

Certainty Equivalence

- MC converges to solution with minimum mean-squared error
  - Best fit to the observed returns

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} \left(G_t^k - V(s_t^k) \right)^2$$

  - In the AB example, $V(A) = 0$
- TD(0) converges to solution of max likelihood Markov model
  - Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^a = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_s^a = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$$

  - In the AB example, $V(A) = 0.75$

Advantages and Disadvantages of MC vs. TD (3)

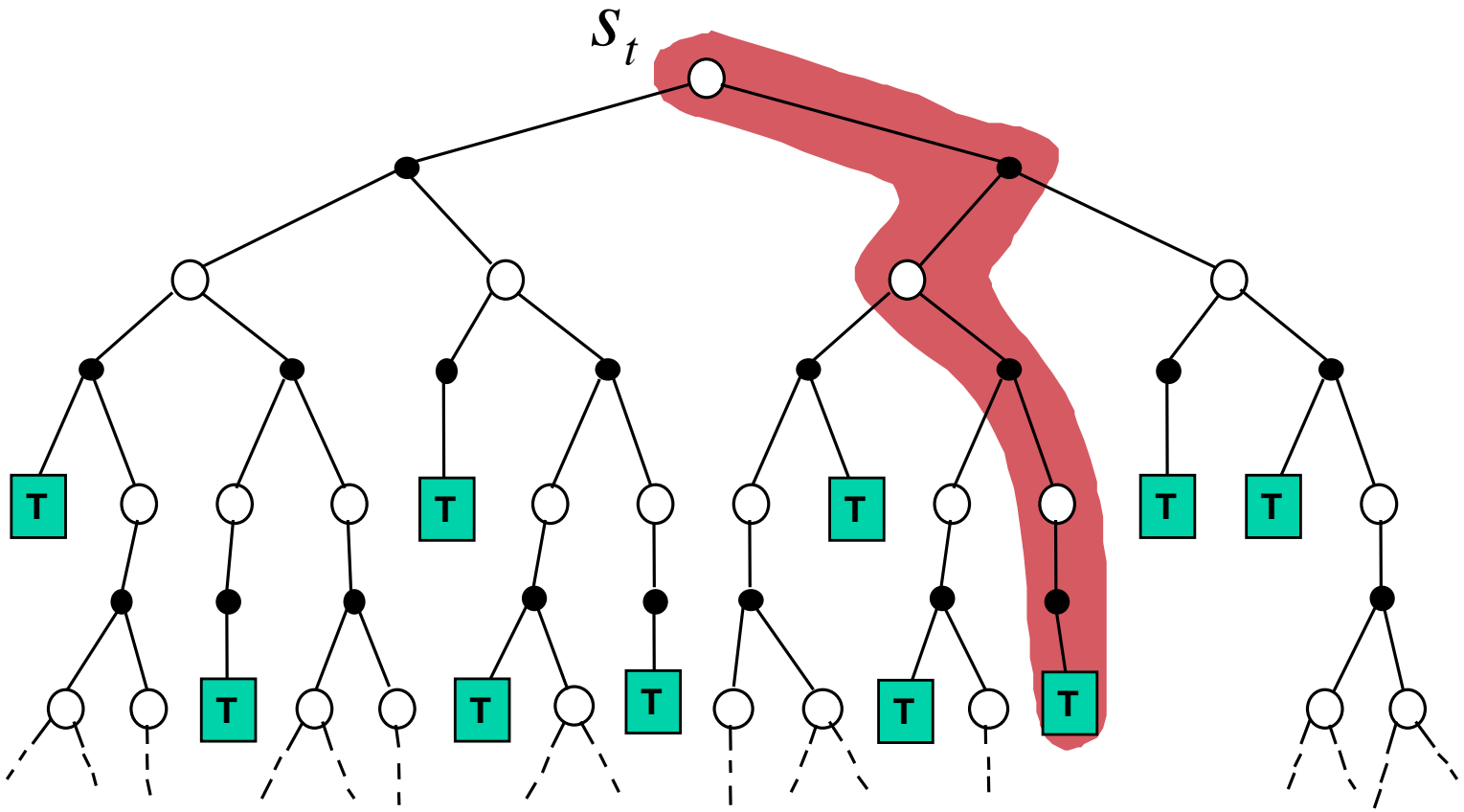
TD는 MP 사용해서 추측
MC는 MP 사용안하고 추측

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

TD는 Markov Model의 likelihood를 maximize함.

Monte-Carlo Backup

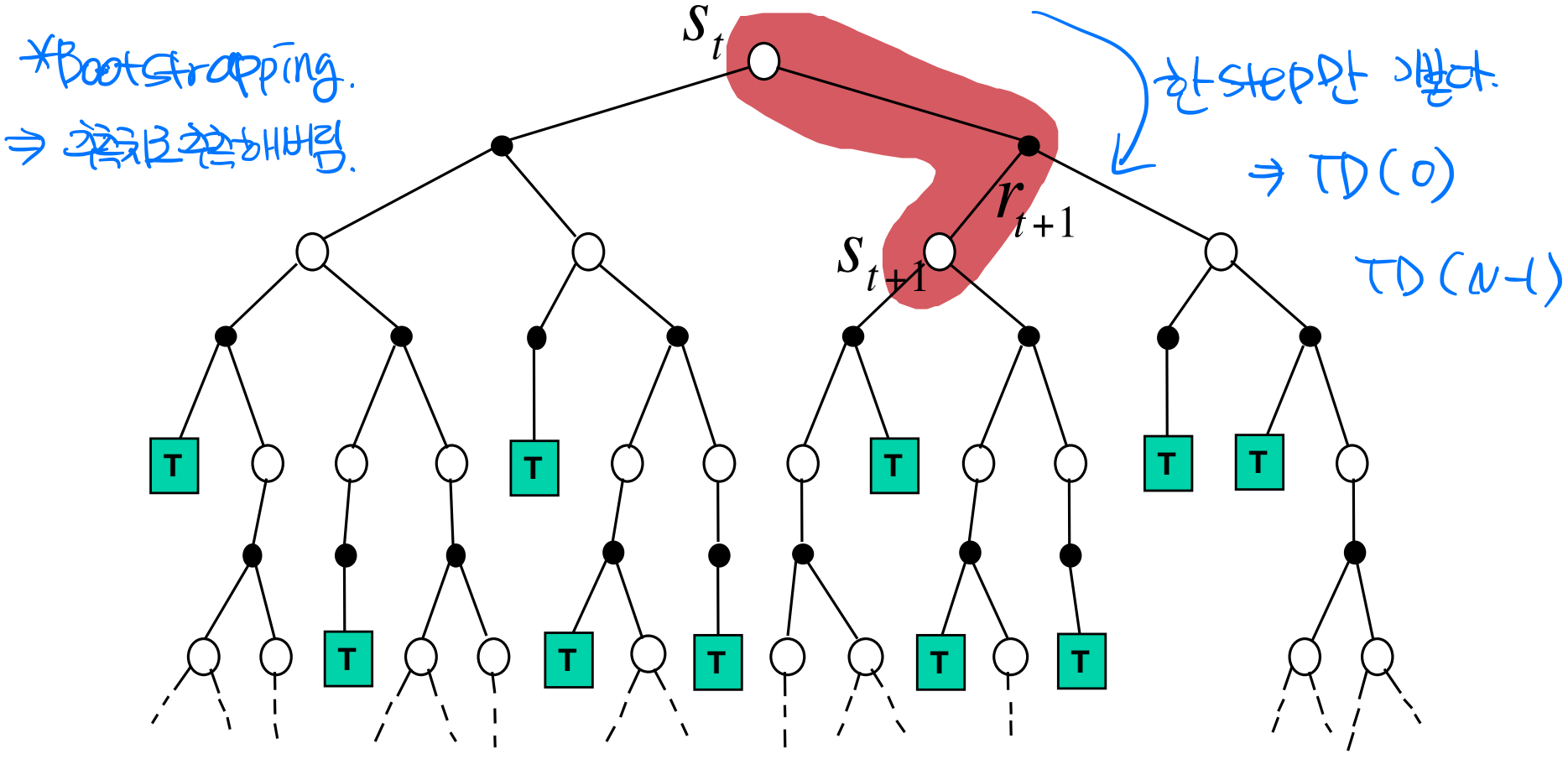
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$



Temporal-Difference Backup

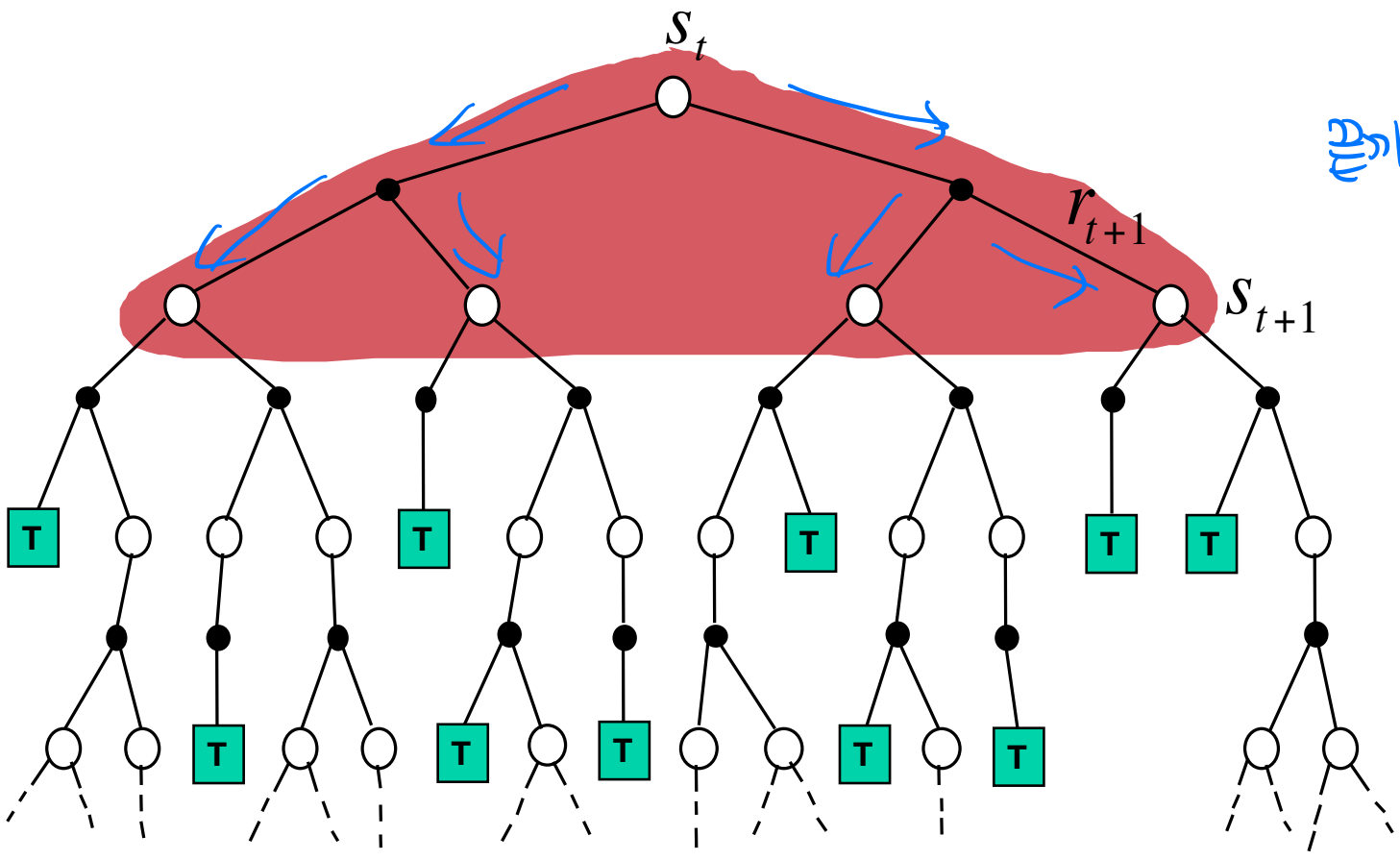
MDP를 알면 hybrid 가능.

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$

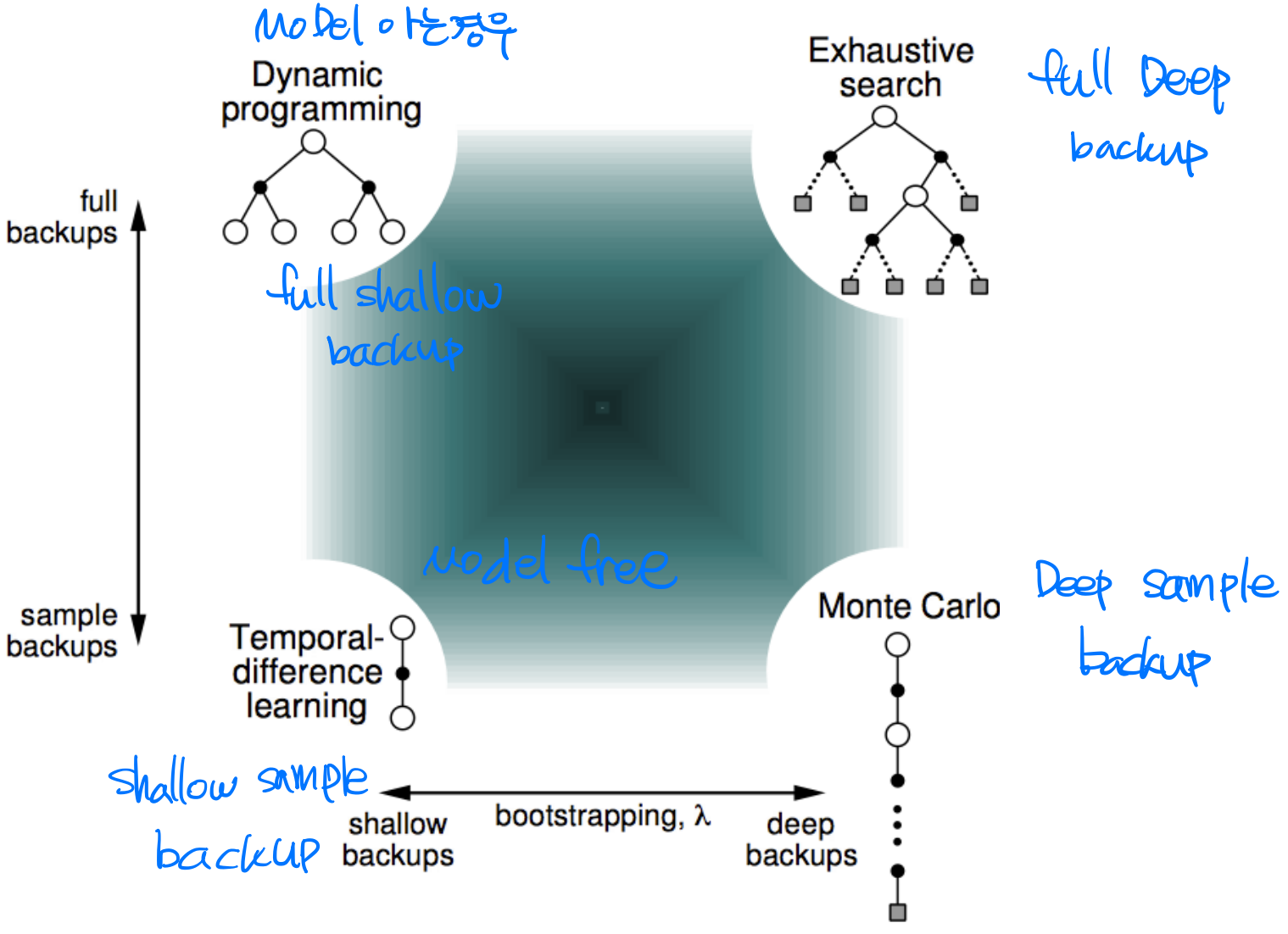


끝까지 가지않는다.

Bootstrapping and Sampling

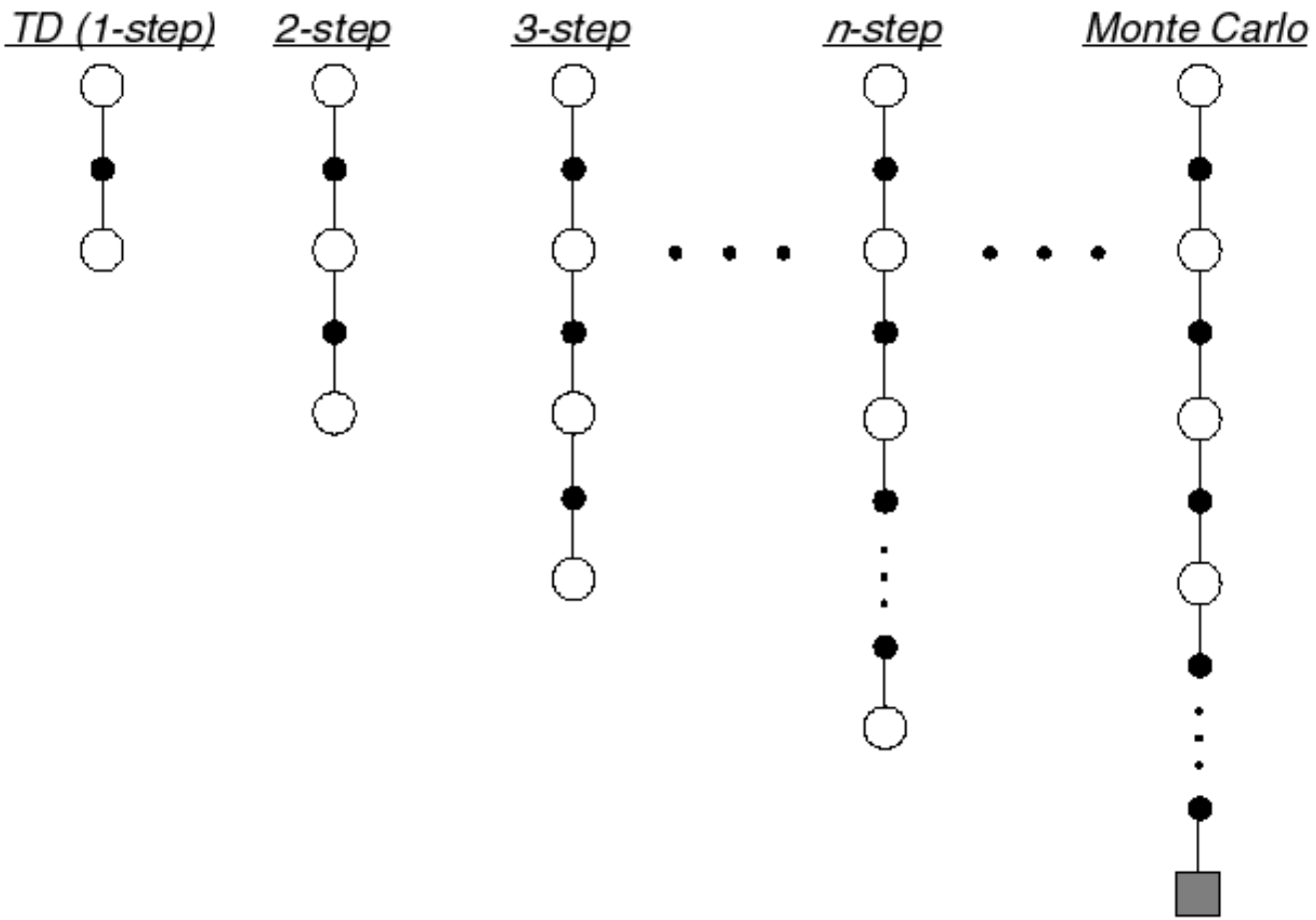
- Bootstrapping: update involves an estimate
  - MC does not bootstrap ⇒ 끝까지 가보니까
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample ⇒ 가능한 action 전부 해보니까 non-sample
  - TD samples

Unified View of Reinforcement Learning



n-Step Prediction

- Let TD target look n steps into the future



끝까지가면 MC가 된다.

n-Step Return

- Consider the following n-step returns for $n = 1, 2, \infty$:

$$
\begin{array}{lll}
n = 1 & (TD) & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\
n = 2 & & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\
\vdots & & \vdots \\
n = \infty & (MC) & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T
\end{array}
$$

- Define the n-step return

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

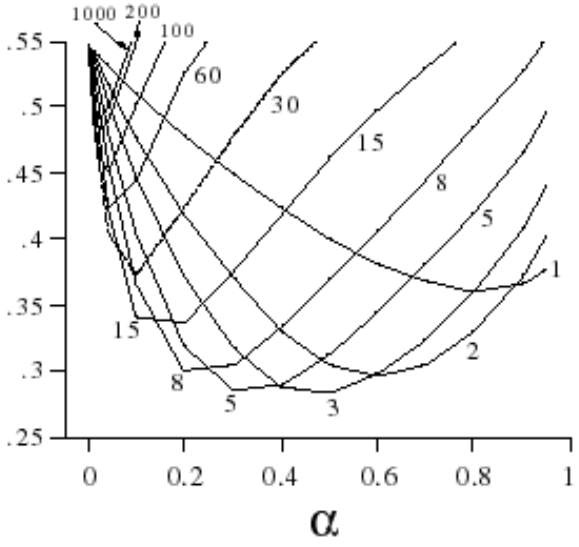
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

n-step TD error

Large Random Walk Example

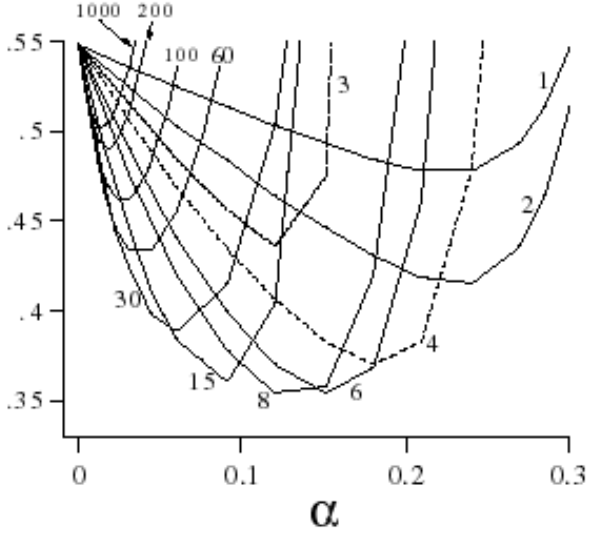
TD(1) ~ TD(1000)

RMS error, averaged over first 10 episodes



ON-LINE → 실시간 update
n-STEP TD

RMS error, averaged over first 10 episodes



OFF-LINE → 다끝난 후 update
n-STEP TD

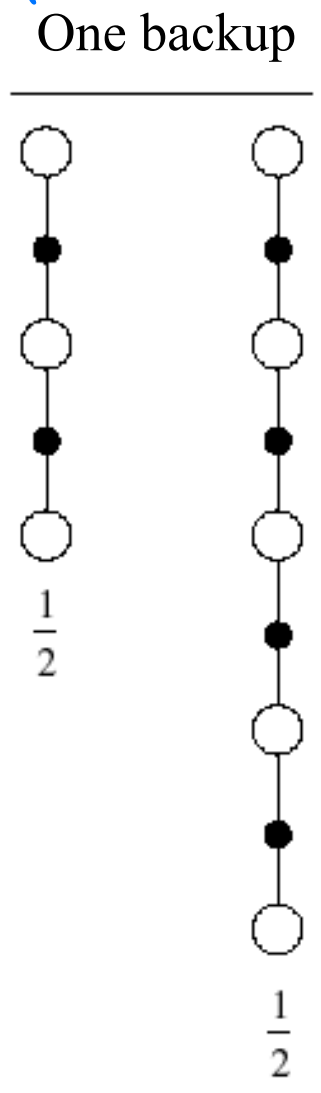
Averaging n-Step Returns

TD(0) ~ MC까지 해서 평균낸다.
↳ 이게 TD(λ)

- We can average n-step returns over different n
- e.g. average the 2-step and 4-step returns

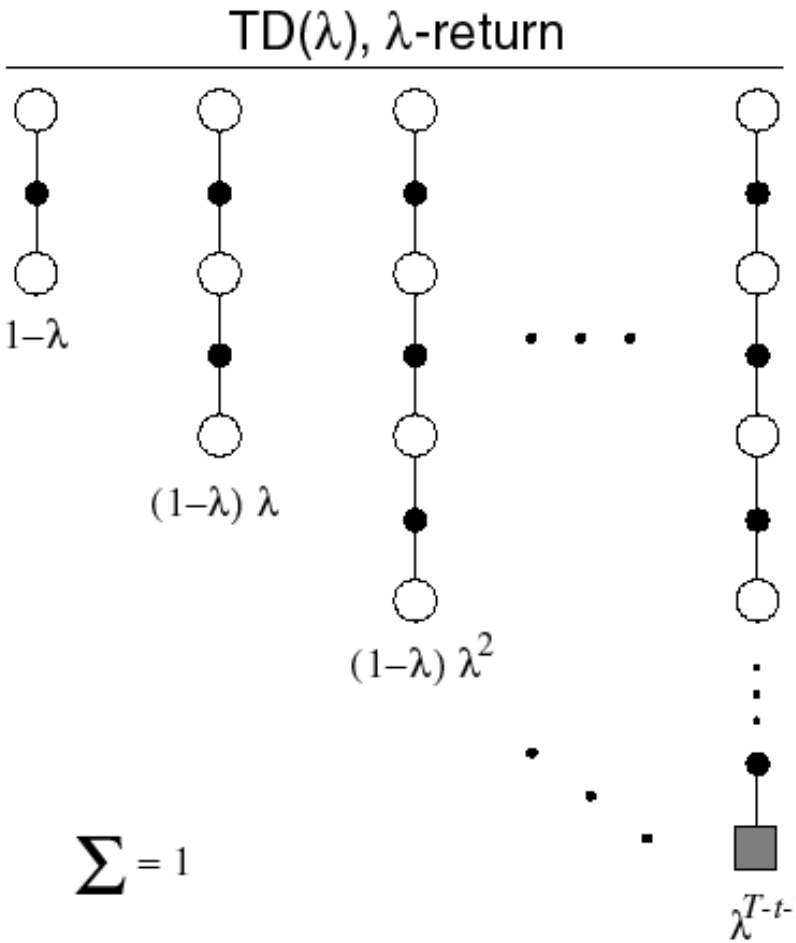
$$\frac{1}{2}G^{(2)} + \frac{1}{2}G^{(4)}$$

- Combines information from two different time-steps
- Can we efficiently combine information from all time-steps?



λ-return

$\lambda = 0 \sim 1$



MC로갈수록 λ가작아짐.

- The λ-*return* G_t^λ combines all n-step returns $G_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

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

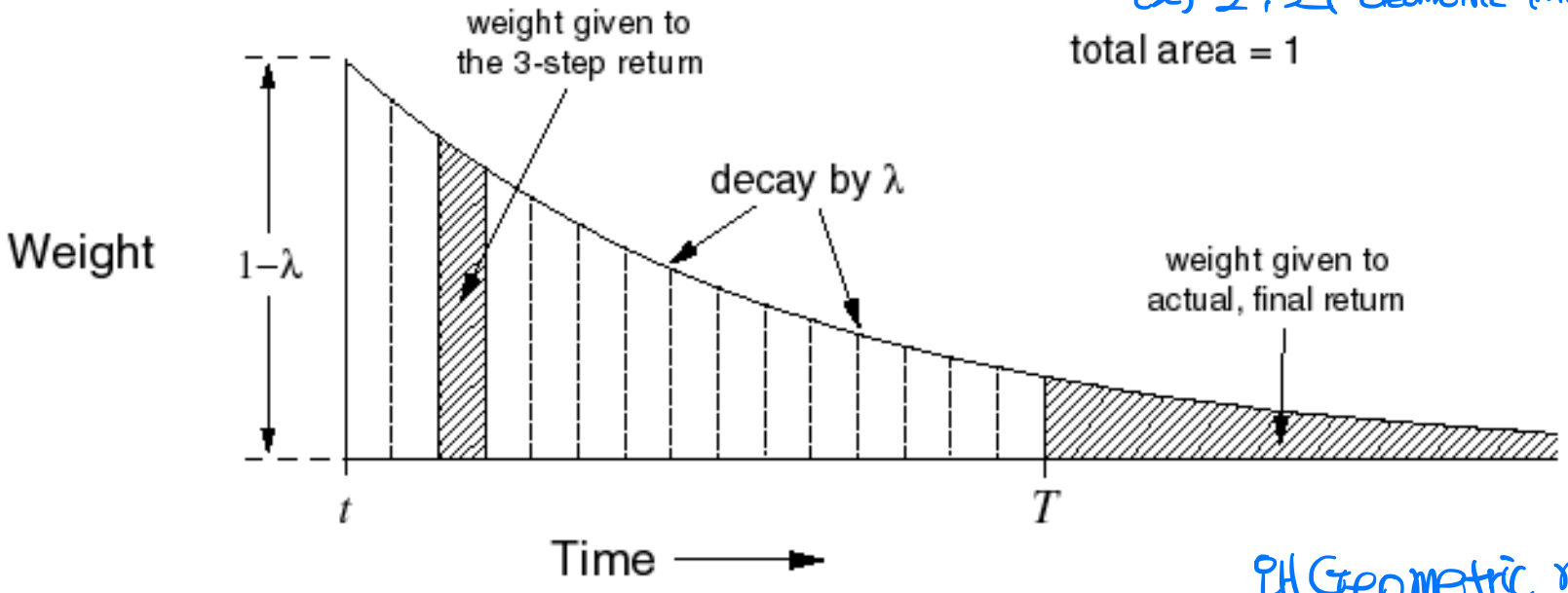
- Forward-view TD(λ)

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^\lambda - V(S_t)\right)$$

TD와MC의 G_t를 다더하는데 $(1-\lambda)\lambda^{n-1}$ 가중치를 사용하는방법.

TD(λ) Weighting Function

Geometric mean
$= (a \times b)^{\frac{1}{2}}, (a \cdot b \cdot c)^{\frac{1}{3}}$
ex) 2, 3의 Geometric mean은 $\sqrt{2 \cdot 3}$

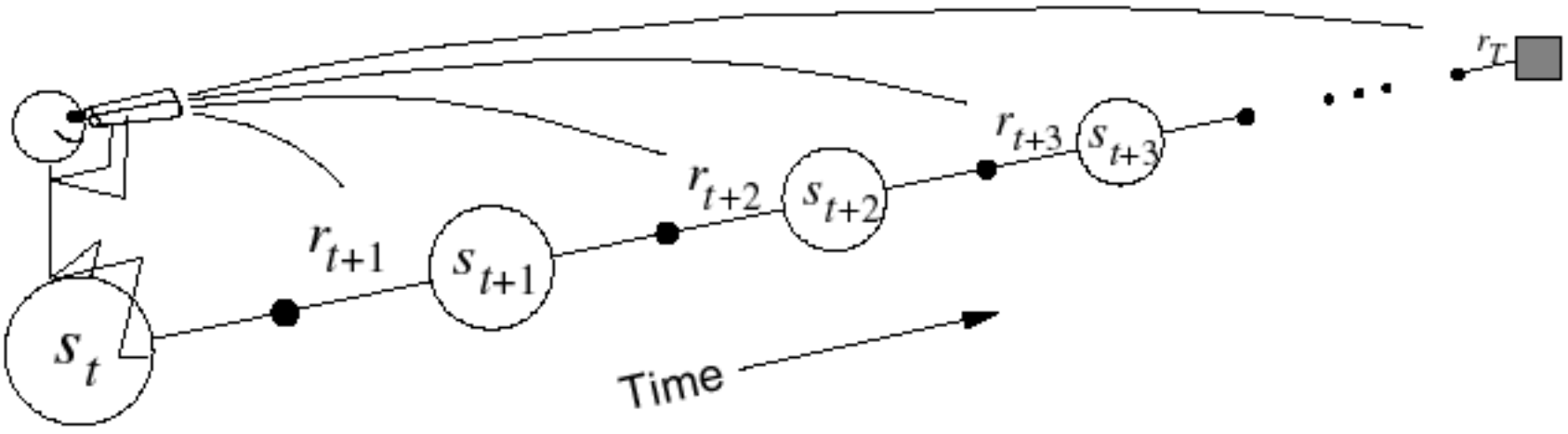


왜 Geometric mean 사용?
컴퓨터 계산적으로 편함.
이전꺼 기억 안하고 계산가능.

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

Forward-view TD(λ)

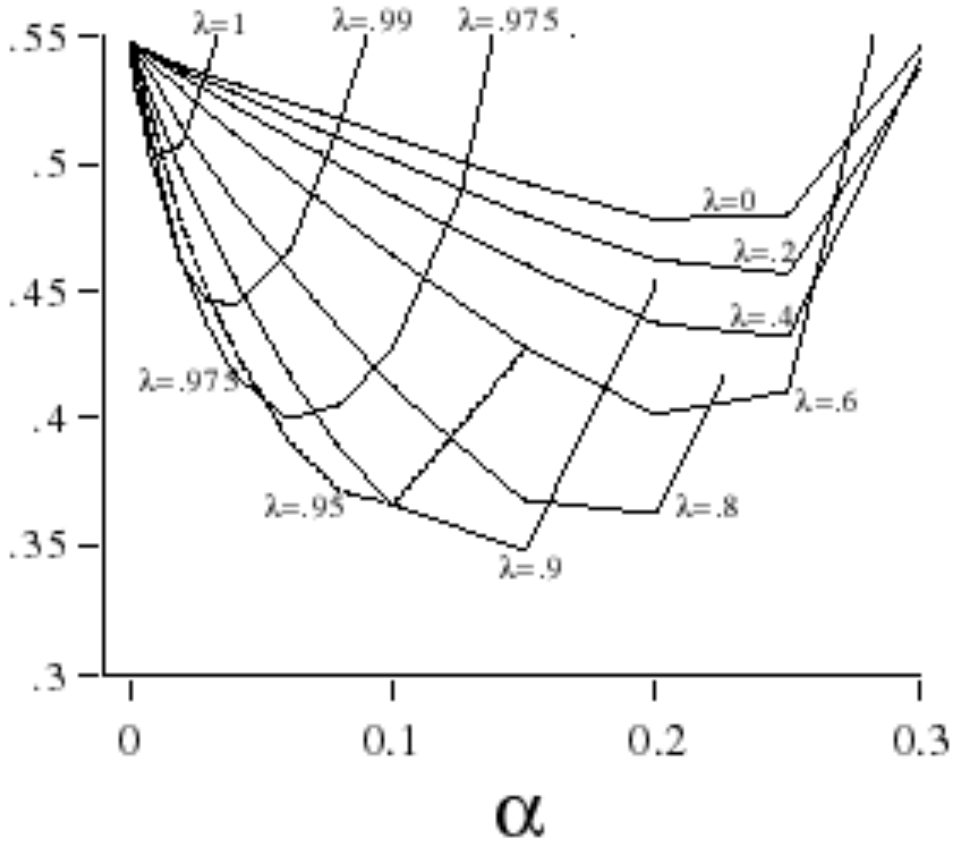
TD(λ)는 어떠한 것을 말하는 것인가



- Update value function towards the λ-return
- Forward-view looks into the future to compute G_t^λ
- Like MC, can only be computed from complete episodes

Forward-View TD(λ) on Large Random Walk

RMS error, averaged over first 10 episodes



OFF-LINE
λ-RETURN

λ이 때가
다 대로다.

Backward View TD(λ)

TD(0)의 장점을 가진 TD(λ)

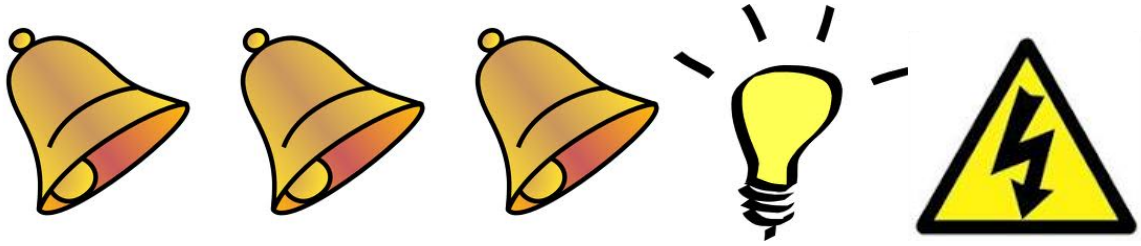
- Forward view provides theory
- Backward view provides mechanism
- Update online, every step, from incomplete sequences

↓
학습하면서 움직인다.

offline은 epi 끝내면 학습후 움직임.

Eligibility Traces

어떤 사건에 책임이 있는 것에 update 많이 해준다.



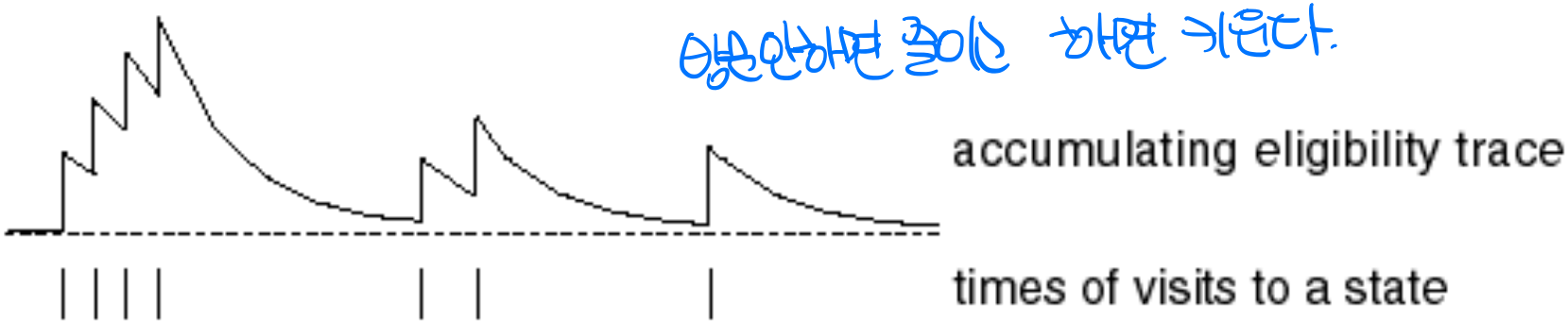
- Credit assignment problem: did bell or light cause shock?
- Frequency heuristic: assign credit to most frequent states
- Recency heuristic: assign credit to most recent states
- *Eligibility traces* combine both heuristics

책임은 둘 다

$$E_0(s) = 0$$

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

행동안하면 줄어들 하면 커진다.

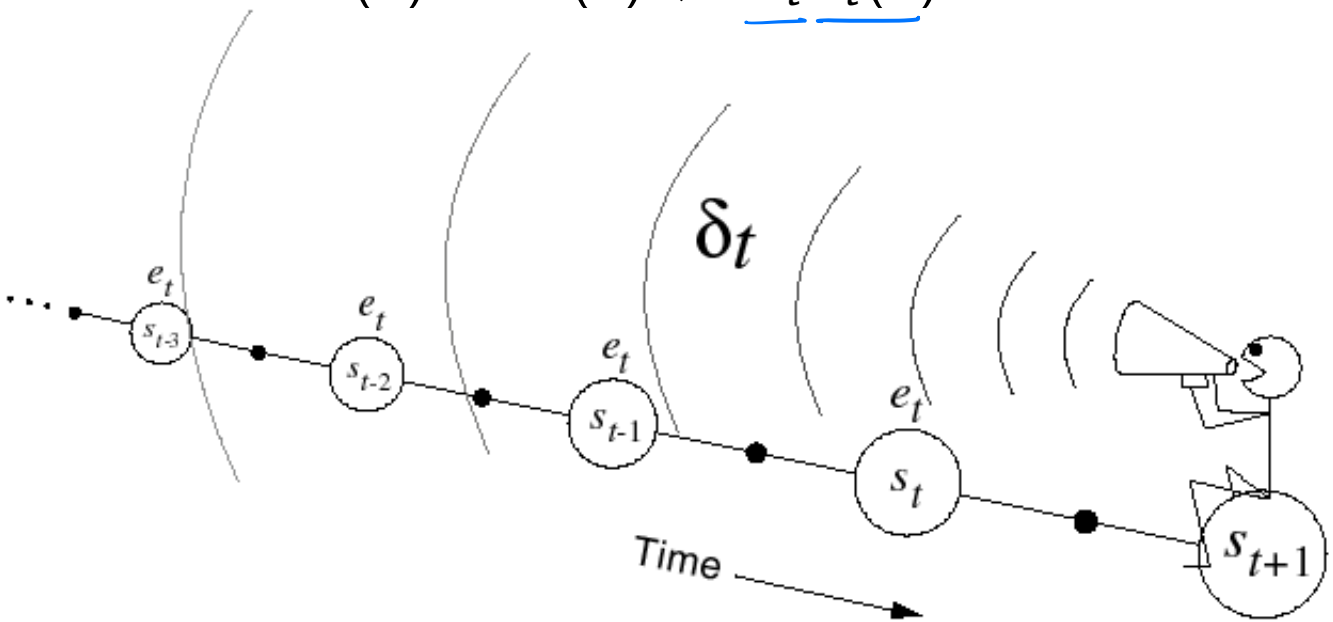


Backward View TD(λ)

- Keep an eligibility trace for every state s
- Update value $V(s)$ for every state s
- In proportion to TD-error δ_t and eligibility trace $E_t(s)$

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

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$



TD(λ) and TD(0)

- When $\lambda = 0$, only current state is updated

$$E_t(s) = \mathbf{1}(S_t = s)$$
$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

- This is exactly equivalent to TD(0) update

$$V(S_t) \leftarrow V(S_t) + \alpha \delta_t$$

TD(λ) and MC

- When $\lambda = 1$, credit is deferred until end of episode
- Consider episodic environments with offline updates
- Over the course of an episode, total update for TD(1) is the same as total update for MC

Theorem

The sum of offline updates is identical for forward-view and backward-view TD(λ)

$$\sum_{t=1}^{T} \alpha \delta_t E_t(s) = \sum_{t=1}^{T} \alpha \left(G_t^{\lambda} - V(S_t) \right) \mathbf{1}(S_t = s)$$

MC and TD(1)

- Consider an episode where s is visited once at time-step k,
- TD(1) eligibility trace discounts time since visit,

$$\begin{aligned} E_t(s) &= \gamma E_{t-1}(s) + \mathbf{1}(S_t = s) \\ &= \begin{cases} 0 & \text{if } t < k \\ \gamma^{t-k} & \text{if } t \geq k \end{cases} \end{aligned}$$

- TD(1) updates accumulate error *online*

$$\sum_{t=1}^{T-1} \alpha \delta_t E_t(s) = \alpha \sum_{t=k}^{T-1} \gamma^{t-k} \delta_t = \alpha \left(G_k - V(S_k) \right)$$

- By end of episode it accumulates total error

$$\delta_k + \gamma \delta_{k+1} + \gamma^2 \delta_{k+2} + \ldots + \gamma^{T-1-k} \delta_{T-1}$$

Telescoping in TD(1)

When $\lambda = 1$, sum of TD errors telescopes into MC error,

$$
\begin{aligned}
& \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \gamma^{T-1-t}\delta_{T-1} \\
&= R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \\
&+ \gamma R_{t+2} + \gamma^2 V(S_{t+2}) - \gamma V(S_{t+1}) \\
&+ \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3}) - \gamma^2 V(S_{t+2}) \\
&\quad \vdots \\
&+ \gamma^{T-1-t} R_T + \gamma^{T-t} V(S_T) - \gamma^{T-1-t} V(S_{T-1}) \\
&= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} ... + \gamma^{T-1-t} R_T - V(S_t) \\
&= G_t - V(S_t)
\end{aligned}
$$

TD(λ) and TD(1)

- TD(1) is roughly equivalent to every-visit Monte-Carlo
- Error is accumulated online, step-by-step
- If value function is only updated offline at end of episode
- Then total update is exactly the same as MC

Telescoping in TD(λ)

For general λ, TD errors also telescope to λ-error, $G_t^\lambda - V(S_t)$

$$
\begin{aligned}
G_t^\lambda - V(S_t) = -V(S_t) \quad &+ \quad (1-\lambda)\lambda^0 \left(R_{t+1} + \gamma V(S_{t+1})\right) \\
&+ \quad (1-\lambda)\lambda^1 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})\right) \\
&+ \quad (1-\lambda)\lambda^2 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})\right) \\
&+ \quad \ldots \\
= -V(S_t) \quad &+ \quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V(S_{t+1}) - \gamma\lambda V(S_{t+1})\right) \\
&+ \quad (\gamma\lambda)^1 \left(R_{t+2} + \gamma V(S_{t+2}) - \gamma\lambda V(S_{t+2})\right) \\
&+ \quad (\gamma\lambda)^2 \left(R_{t+3} + \gamma V(S_{t+3}) - \gamma\lambda V(S_{t+3})\right) \\
&+ \quad \ldots \\
= \quad &\quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \\
&+ \quad (\gamma\lambda)^1 \left(R_{t+2} + \gamma V(S_{t+2}) - V(S_{t+1})\right) \\
&+ \quad (\gamma\lambda)^2 \left(R_{t+3} + \gamma V(S_{t+3}) - V(S_{t+2})\right) \\
&+ \quad \ldots \\
= \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \ldots &
\end{aligned}
$$

Forwards and Backwards TD(λ)

- Consider an episode where s is visited once at time-step k,
- TD(λ) eligibility trace discounts time since visit,

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$
$$= \begin{cases} 0 & \text{if } t < k \\ (\gamma\lambda)^{t-k} & \text{if } t \geq k \end{cases}$$

- Backward TD(λ) updates accumulate error *online*

$$\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \alpha \sum_{t=k}^{T} (\gamma\lambda)^{t-k}\delta_t = \alpha\left(G_k^\lambda - V(S_k)\right)$$

- By end of episode it accumulates total error for λ-return
- For multiple visits to s, $E_t(s)$ accumulates many errors

Offline Equivalence of Forward and Backward TD

Offline updates

- Updates are accumulated within episode
- but applied in batch at the end of episode

Onine Equivalence of Forward and Backward TD

Online updates

- TD(λ) updates are applied online at each step within episode
- Forward and backward-view TD(λ) are slightly different
- NEW: Exact online TD(λ) achieves perfect equivalence
- By using a slightly different form of eligibility trace
- Sutton and von Seijen, ICML 2014

Summary of Forward and Backward TD(λ)

Offline updates	$\lambda = 0$	$\lambda \in (0, 1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\Vert$	$\Vert$	$\Vert$
Forward view	TD(0)	Forward TD(λ)	MC
Online updates	$\lambda = 0$	$\lambda \in (0, 1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\Vert$	$\nparallel$	$\nparallel$
Forward view	TD(0)	Forward TD(λ)	MC
	$\Vert$	$\Vert$	$\Vert$
Exact Online	TD(0)	Exact Online TD(λ)	Exact Online TD(1)

$=$ here indicates equivalence in total update at end of episode.