5. Model-free Prediction

2019 Fall

Yusung Kim
yskim525@skku.edu

Review

- Return is total discounted sum of rewards from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Immediate reward   Discount sum of Future rewards

- $v_\pi(s)$ is expected return from starting in state s under policy π

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

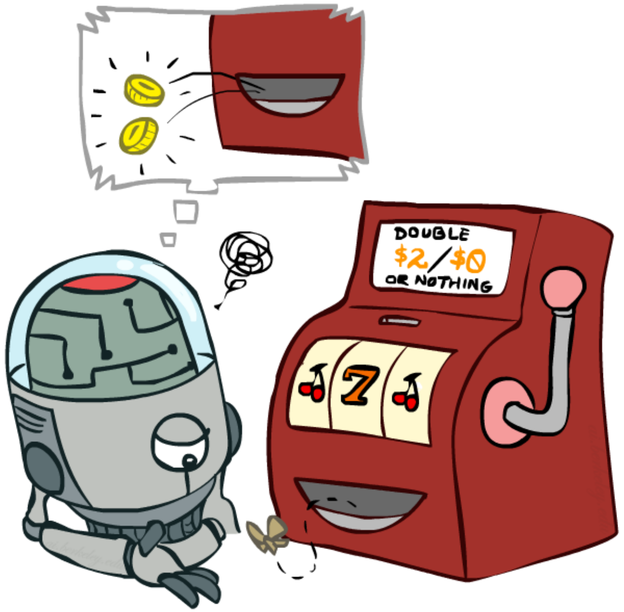
- $q_\pi(s, a)$ is expected return from starting in state s, taking action a under policy π

$$q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

Model-free Reinforcement Learning

- Estimating the expected return of a particular policy without true MDP models.

MDP를 모르는데 그냥 환경이 던져짐.



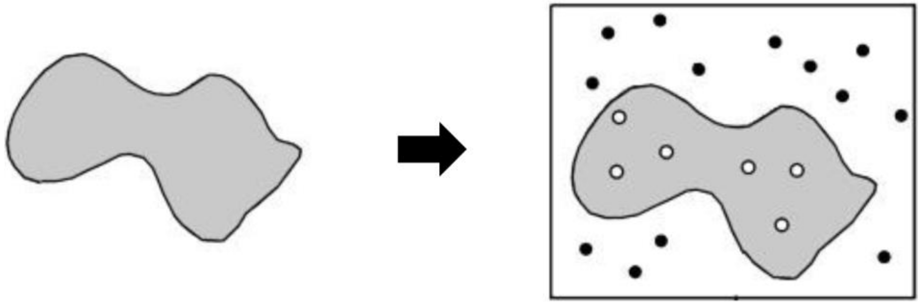
Model-free Reinforcement Learning

- **Model-free prediction (evaluation)**
  - Estimate the value function of an unknown MDP
  - How good is this given policy ?
- **Model-free control (improvement)**
  - Optimize the value function of an unknown MDP
  - How can we learn a better policy ?

What are Monte-Carlo Methods?

실제 값들을 통해서 추정하는 것.
policy를 따라서 계속 해봄.

- A class of computational algorithms that rely on repeated random sampling to obtain numerical results.
- How to measure the area of an irregular shape?



Monte-Carlo Reinforcement Learning

- 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
- Only for episodic MDPs (all episodes must terminate)

끝까지 해보고 return값을 구함.
그 return들의 평균을 낸 것이 value.
모든 에피소드가 끝나야만 정할 수 있음.

Monte-Carlo Policy Evaluation

- Goal: learn $v_\pi(s)$ from episodes of experience under policy π

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

- Recall that the return is the total discounted reward:

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

- Recall that the value function is the `expected return`:

$$V_\pi(s) = \mathbb{E}_\pi[\, G_t \mid S_t = s \,]$$

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

First-Visit Monte-Carlo Policy Evaluation

처음 방문한 것만 count를 올려줌.

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

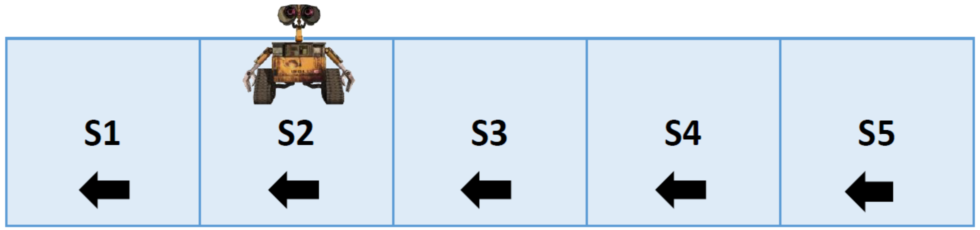
Every-Visit Monte-Carlo Policy Evaluation

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

모든 state를 방문해야만 함.

Example: Mars Rover

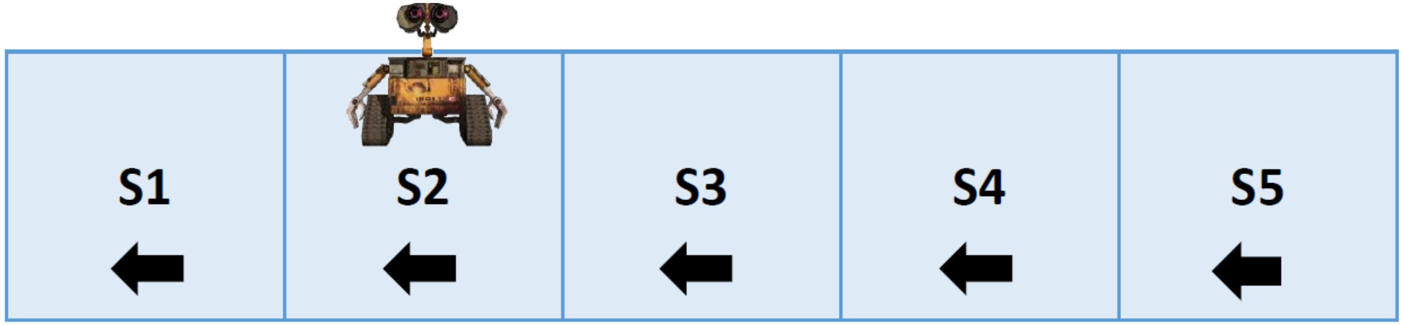
- We do NOT know the model such as state transition prob. `p(S'| S, A)`
- Policy : only "move left" in all states
- Reward : -1 every movement, +1 arriving at S1
- Discount factor $\gamma = 0.5$



$$\cdots \; S_t \xrightarrow{A_t} R_{t+1}, S_{t+1} \xrightarrow{A_{t+1}} R_{t+2}, S_{t+2} \xrightarrow{A_{t+2}} R_{t+3}, S_{t+3} \xrightarrow{A_{t+3}} \cdots$$

Example: Mars Rover

- Case of First-Visit MC



- Sample episodes
  - S2, -1, S3, -1, S2, +1, S1

$$S2: N(S2) = 1, \qquad V(S2) = \frac{(-1 + 0.5 * -1 + 0.5^2 * 1)}{1} = -1.25$$

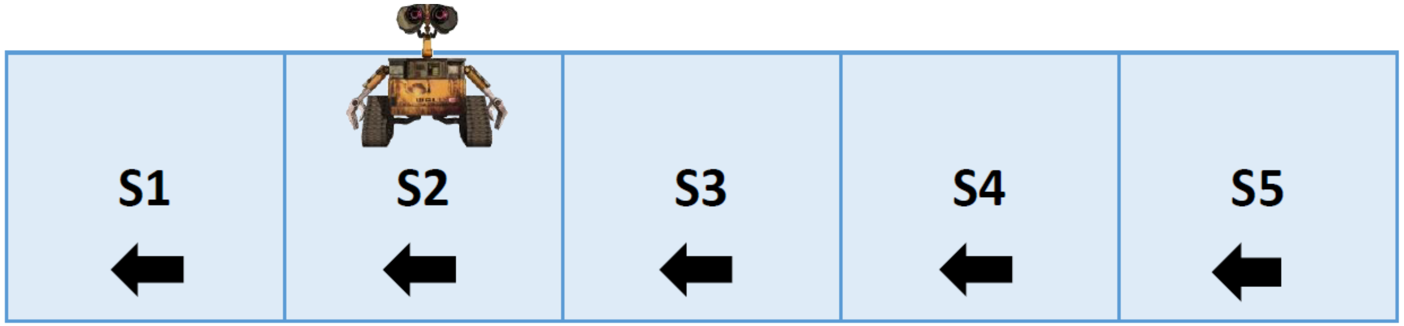
$$S3: N(S3) = 1, \qquad V(S3) = \frac{(-1 + 0.5 * 1)}{1} = -0.5$$

  - S2, +1, S1

$$S2: N(S2) = 2, \qquad V(S2) = \frac{(-1.25 + 1)}{2} = -0.125$$

Example: Mars Rover

- Case of Every-Visit MC
- Sample episodes
  - S2, -1, S3, -1, S2, +1, S1
  - S2, +1, S1



$S2: N(S2) = 3, V(S2) = (-1.25 + 1 + 1)/3 = 0.25$

$S3: N(S3) = 1, V(S3) = -0.5/1 = -0.5$

Incremental Mean

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

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

mean을 구하려면 에피소드마다 결과값을 저장하고 있어야 하는데
incremental mean을 사용하면 저장하고 있지 않고 그때마다 구해주면 된다.

Incremental Monte-Carlo Update

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, \dots, S_k \sim \pi$
- For each state S_t with return G_t

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

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

G-V = error
error만큼 update해주는 것.

- In non-stationary problems, it can be useful to track a running mean

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

  - $\alpha > \frac{1}{N(s)}$: forget older data

non-stationary problem : MDP가 조금씩 계속 바뀌는 것.
과거의 data는 잊고 최신 것들로 채움.

Temporal-Difference Learning

- 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

  에피소드가 안 끝나도 배울 수 있음.
- TD updates a guess towards a guess

> *"If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning."*
>
> *by Sutton and Barto 2017*

Monte Carlo vs. Temporal Difference

- Learn $v_\pi(s)$ from episodes of experience under policy π
- Incremental Monte-Carlo: $V(S_t) \leftarrow V(S_t) + \alpha\,(\,G_t - V(S_t)\,)$ G의 방향으로 update
- Temporal-difference learning algorithm
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$ 한 step 더 가서 예측한 값.
    한 step 더 간 것이 조금 더 정확할 것.

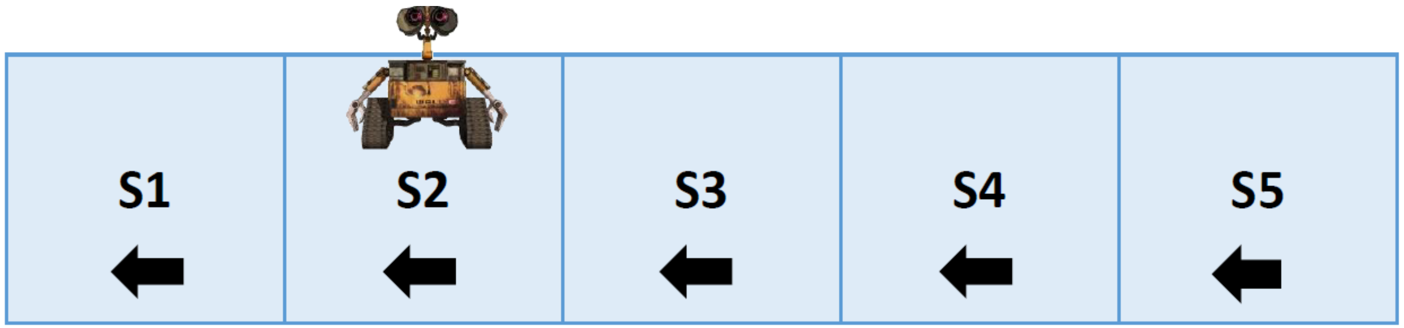
$$V(S_t) \leftarrow V(S_t) + \alpha\,(\,R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\,)$$

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

한 step 더 간 예측치로 현재의 예측치를 update.

Example : TD Policy Evaluation

- Sample episodes
  - S2, -1, S3, -1, S2, +1,



$$S2: V(S2) \leftarrow V(S2) + \alpha(-1 + \gamma V(S3) - V(S2))$$

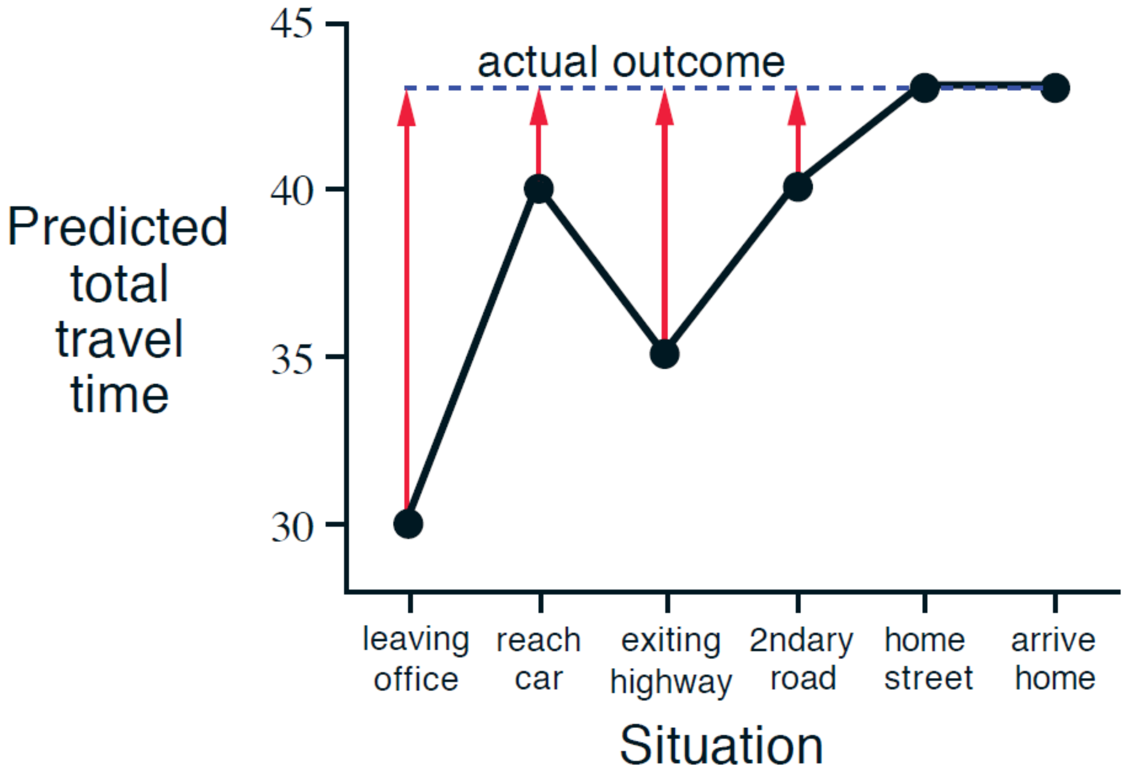
$$S3: V(S3) \leftarrow V(S3) + \alpha(-1 + \gamma V(S2) - V(S3))$$

$$S2: V(S2) \leftarrow V(S2) + \alpha(+1 + \gamma V(S1) - V(S2))$$

Driving Home Example

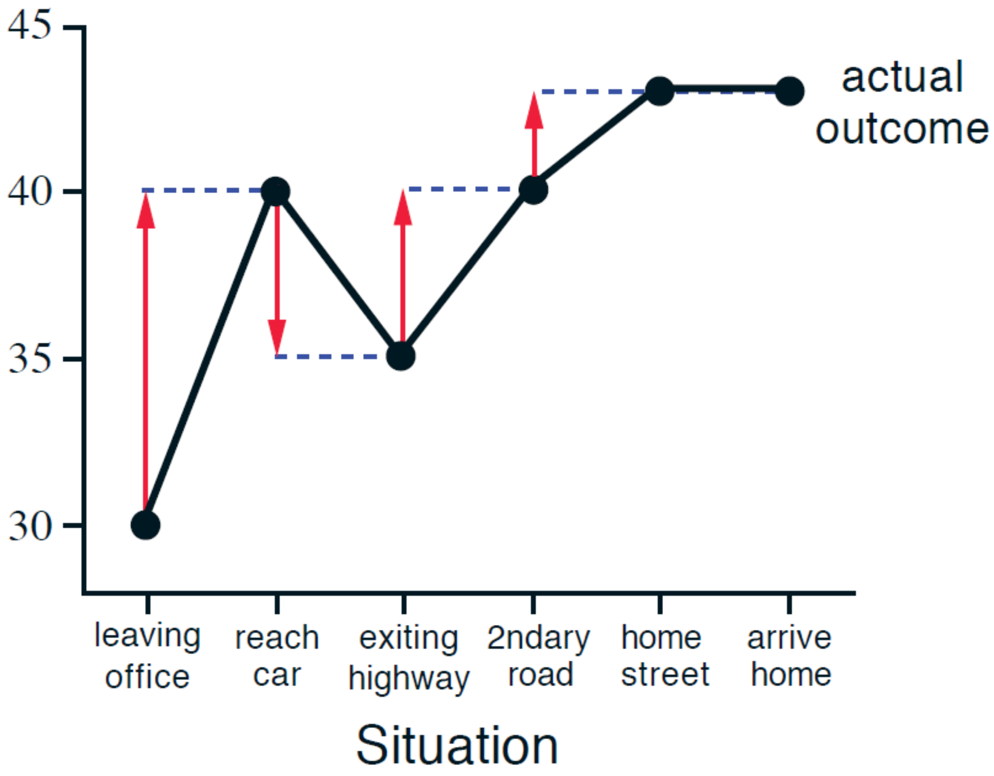
State	실제 걸린 시간 *Elapsed Time (minutes)*	도착 예정 시간 *Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Driving Home Example



Monte Carlo Method

전부 43으로 update됨.



Temporal Difference Method

MC vs. TD

- MC can only learn from **complete** sequences
  - MC must wait until end of episode
  - MC only works for episodic (**terminating**) environments
- TD can learn from **incomplete** sequences
  - TD can learn before knowing the final outcome
  - TD can learn online after every step
  - TD works in **continuing** (non-terminating) environments

Bias/Variance Trade-Off

- The bias is an error from erroneous assumptions in the learning algorithm
- The variance is an error from sensitivity to small fluctuations in the training set.

Bias : 얼마나 편향되어있는가
Variance : 평균으로부터 얼마나 퍼져있는가

Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + \ \dots$ is unbiased estimate of $v_\pi(s_t)$.
- True TD target is unbiased estimate of $v_\pi(s_t)$.

한 step사이의 random성은 적음.

MC는 게임 끝의 값을 가지고 update하는데 게임이 끝날 때까지의 random성이 크다.

- Usually, TD target is biased estimate of $v_\pi(s_t)$.
- TD target is much lower variance than the return.

MC vs. TD again

- MC has high variance, zero bias
  - Good convergence properties
    (even with function approximation)
  - Not very sensitive to initial value
  - Very simple to understand and use

- TD has low variance, some bias
  - Usually more efficient than MC
  - TD converges to $v_\pi(s)$
    (but not always with function approximation)
  - More sensitive to initial value

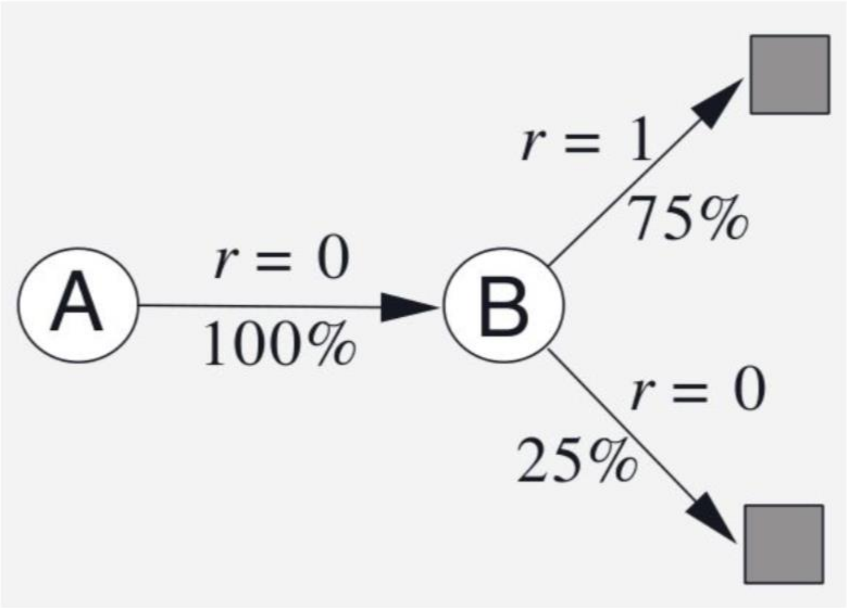
Batch MC and TD

- Batch (Offline) solution for finite dataset
  - Given set of K episodes
  - Repeatedly sample an episode from $k \in [1, K]$
  - Apply MC or TD to the sampled episode
- What do MC and TD converge to?

AB Example:

- Two states A, B with $\gamma = 1$
- Given 8 episodes of experience:
  - $A, 0,\ B, 0$
  - $B, 1$
  - $B, 1$
  - $B, 1$
  - $B, 0$
  - $B, 1$
  - $B, 1$
  - $B, 1$
- What are $V(A), V(B)$?

MC V(A), TD V(A): 0, 0.75

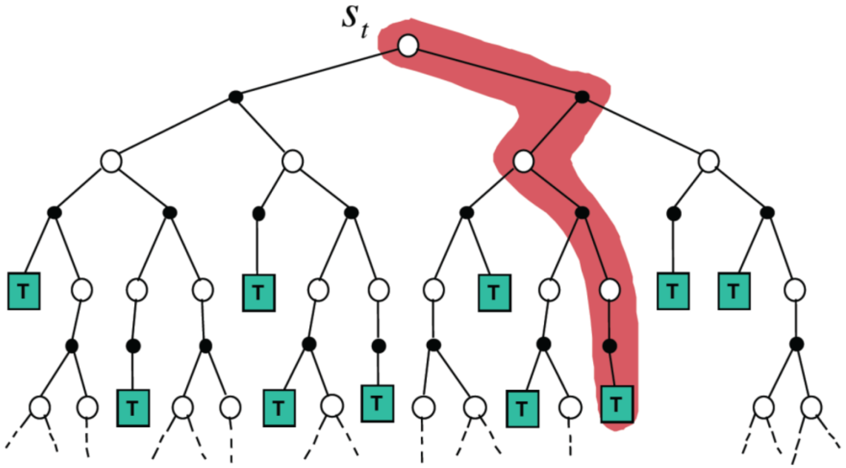


MC vs. TD

- In simplest TD, use (s, a, r, s') once to update $V(s)$
  - $O(1)$ operation per update
  - In an episode of length L, $O(L)$
- In MC have to wait till episode finishes, then also $O(L)$
- TD exploits Markov structure
  - If in Markov domain, leveraging this is helpful
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

DP vs. MC vs. TD

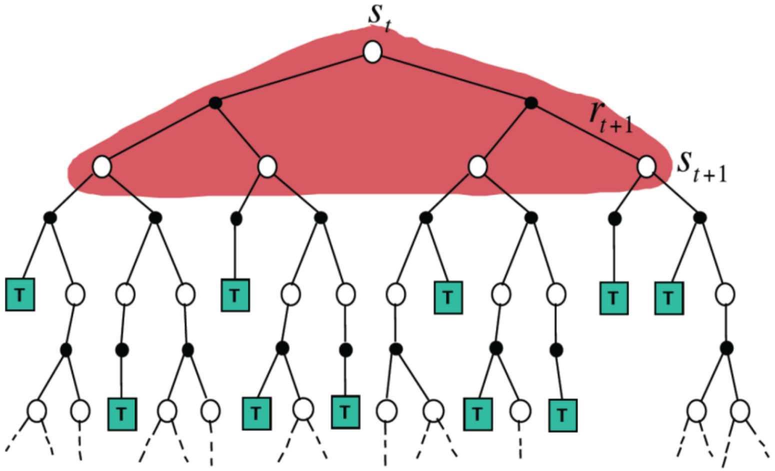
$V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$



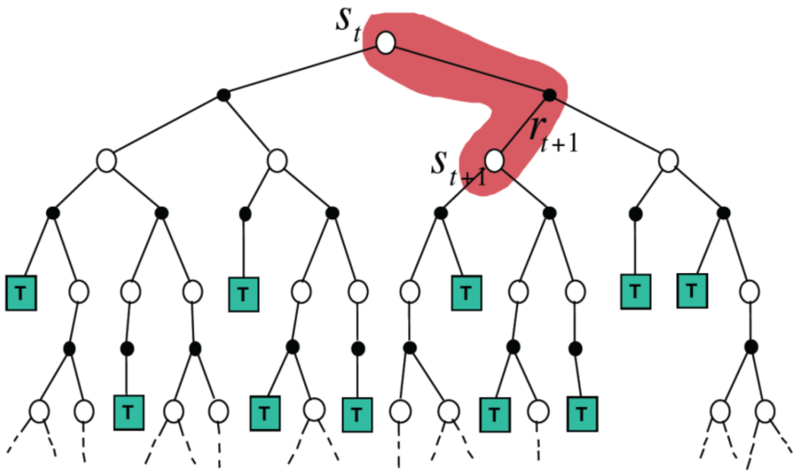
Monte Carlo

Dynamic Programming

$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$



$V(s_t) \leftarrow V(s_t) + \alpha(R_{t+1} + \gamma V(s_{t+1}) - V(s_t))$



Temporal Difference

Sampling and Bootstrapping

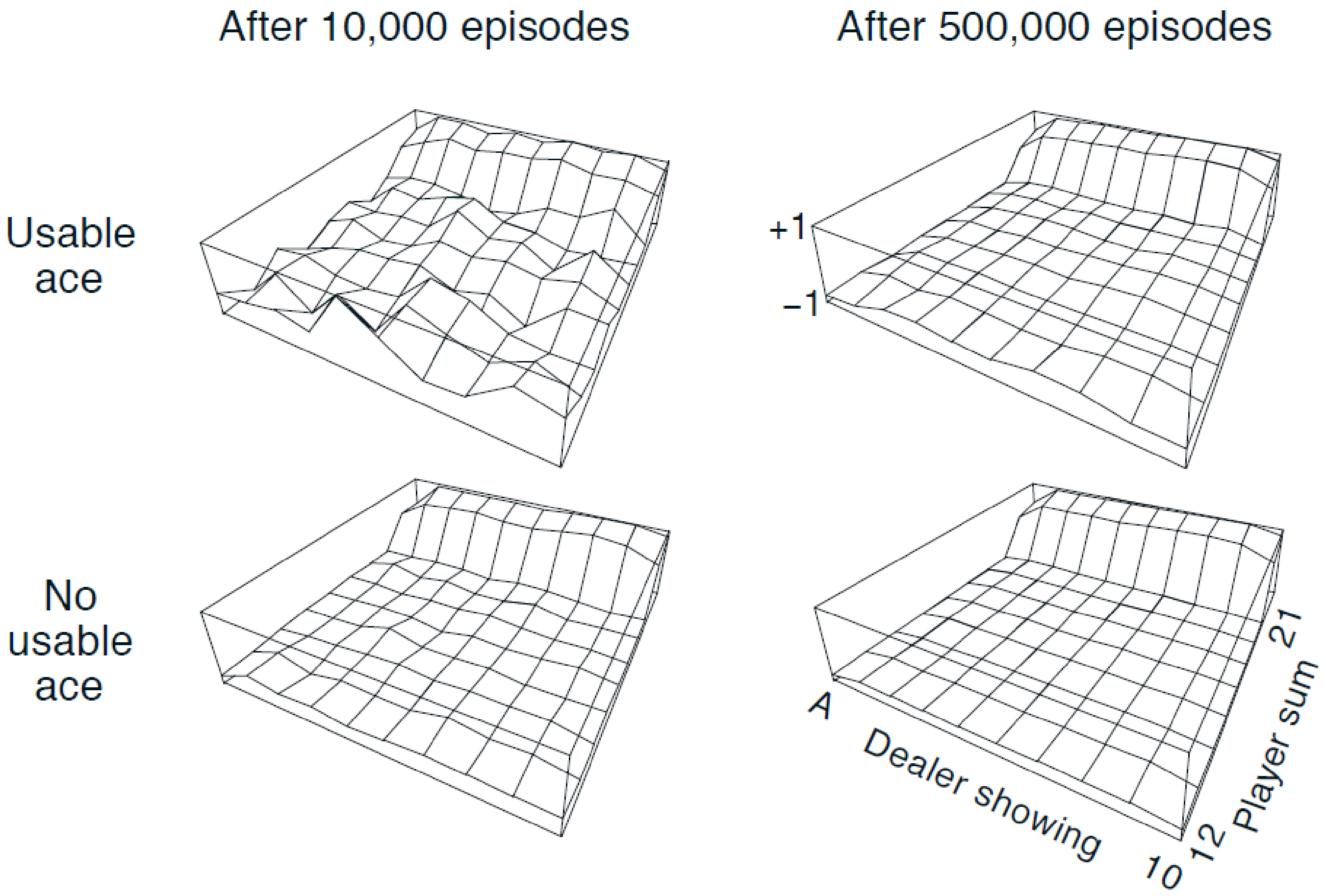
- Sampling: gather information from episodes of experience
  - DP does NOT sample
  - MC samples
  - TD samples

- Bootstrapping: update estimates on the basis of other estimates
  - DP bootstraps
  - MC does NOT bootstrap
  - DP bootstraps

Blackjack Example

- States (280 of them):
  - Current sum ($4 \sim 21$)
  - Dealer's showing card ($ace \sim 10$)
  - Do I have a "useable" ace? ($yes - no$)
- Actions
  - stick: Stop receiving cards (and terminate)
  - hit: Take another card (no replacement)
- Transitions: automatically hit if sum of cards < 12
- 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 hit:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise

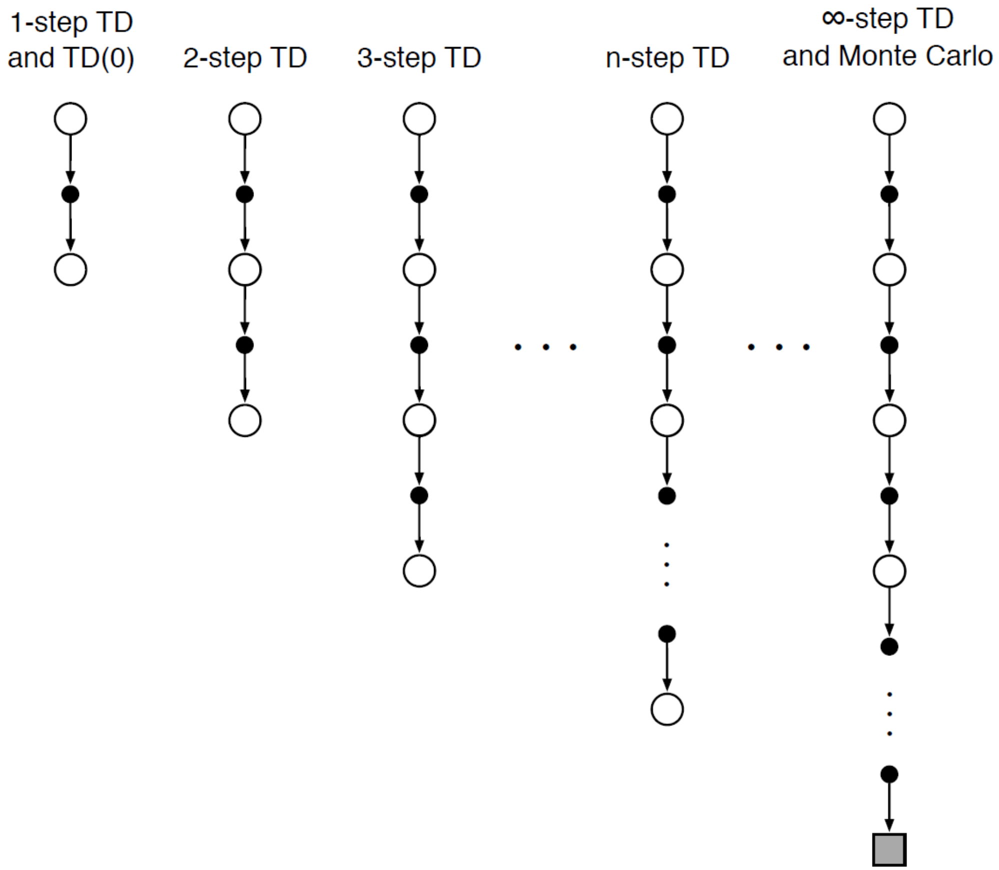
Blackjack Value Function after Prediction



for policy that sticks only on 20 or 21

n-Step Prediction

- Let TD target look n steps into the future



n-Step Return

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

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

- Dene the n-step return

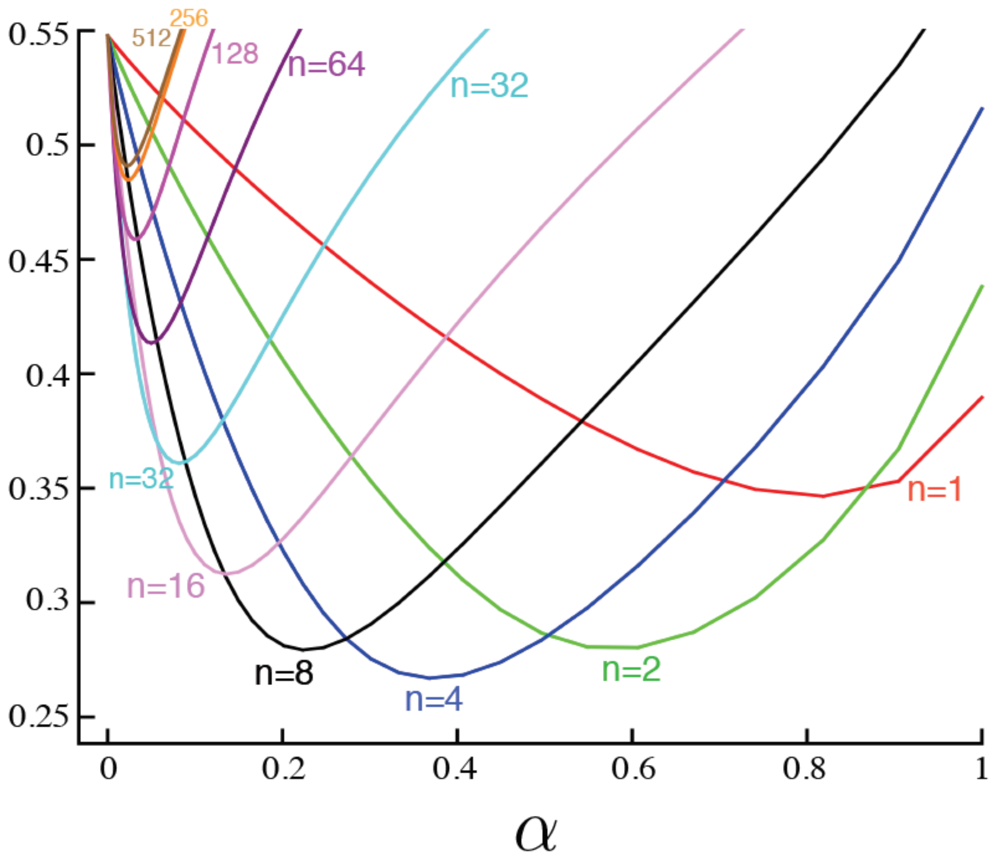
$$
G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})
$$

- n-step temporal-difference learning

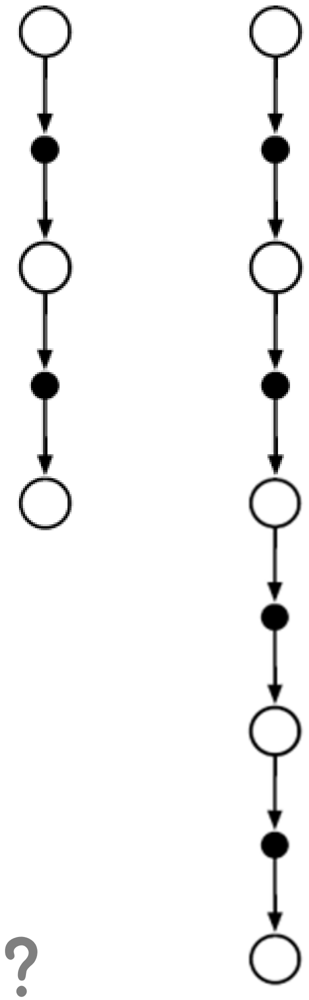
$$
V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)
$$

n-Step at Random Walk Example

Average
RMS error
over 19 states
and first 10
episodes



Averaging n-Step Returns



- 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

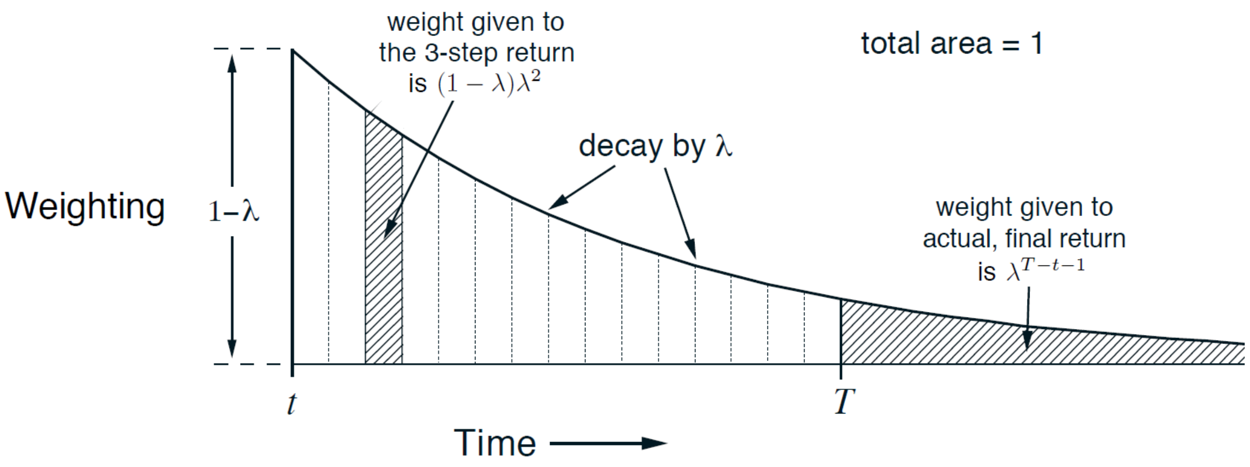
- The TD(λ) as one particular way of averaging n-step updates.
  - Each weighted proportionally to λ^{n-1}(where $\lambda \in [0,1]$)
  - λ-return:

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

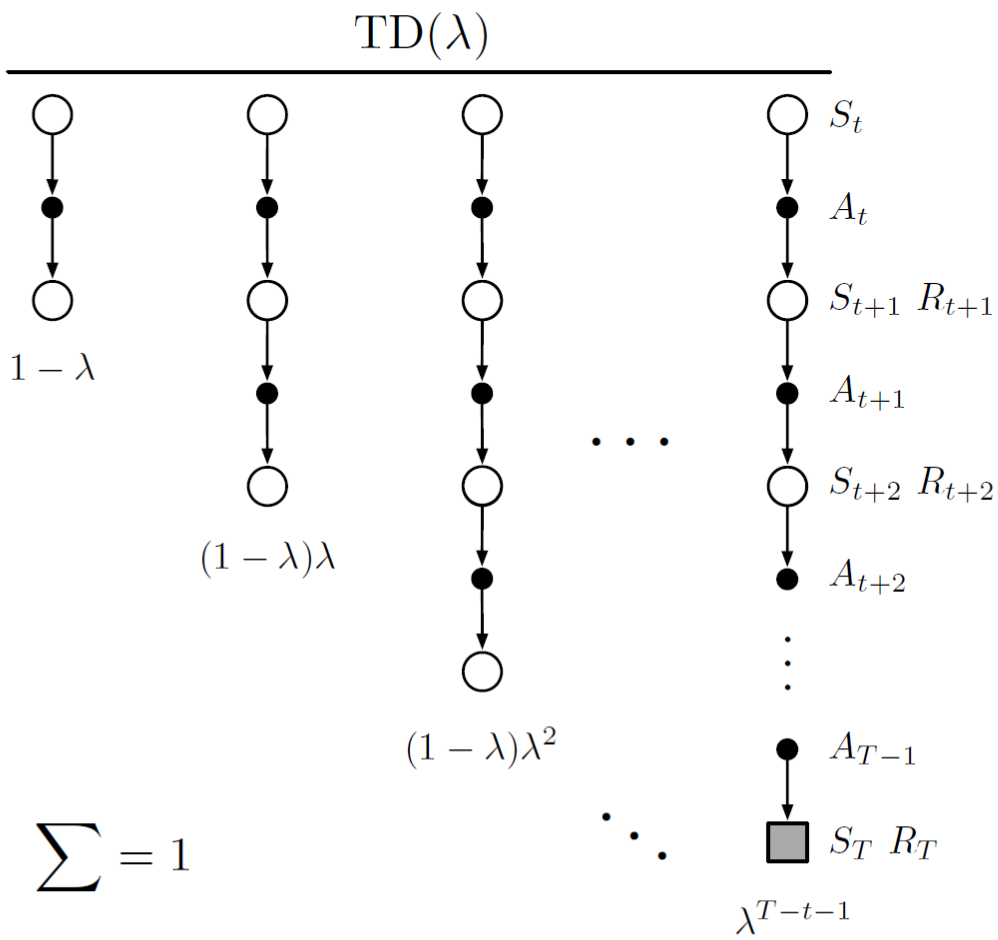
  - Backup using λ-return:

$$V(s_t) = V(s_t) + \alpha[G_t^\lambda - V_t(s_t)]$$

λ-return



weight given to the 3-step return is $(1-\lambda)\lambda^2$
total area = 1
decay by λ
Weighting
1–λ
weight given to actual, final return is λ^{T-t-1}
t
T
Time



TD(λ)
S_t
A_t
S_{t+1} R_{t+1}
A_{t+1}
S_{t+2} R_{t+2}
A_{t+2}
A_{T-1}
S_T R_T
$1-\lambda$
$(1-\lambda)\lambda$
$(1-\lambda)\lambda^2$
λ^{T-t-1}
$\sum = 1$

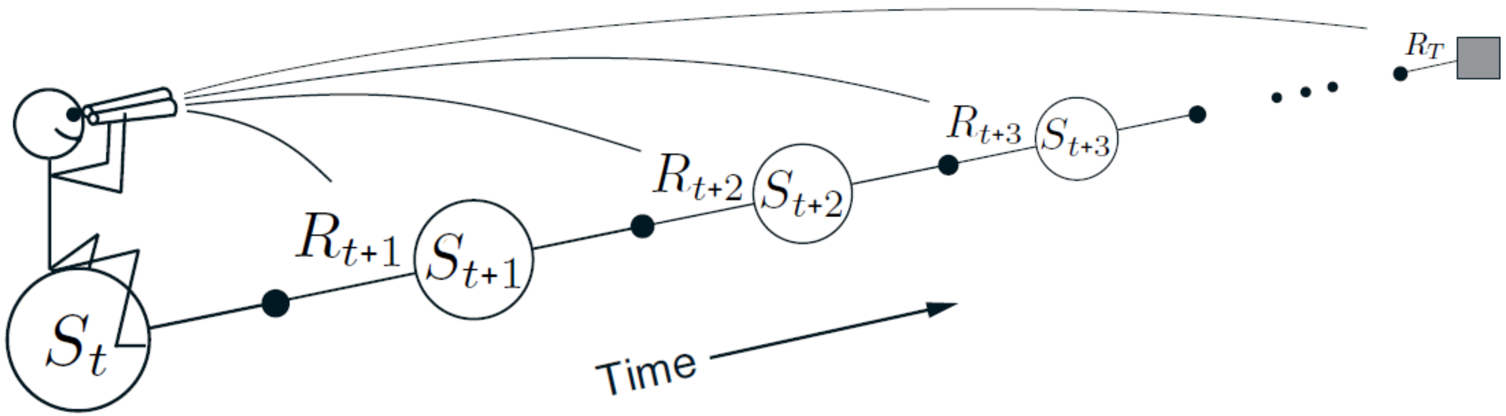
Relation to TD and MC

- if $\lambda = 0$, you get one-step TD, TD(0)
- if $\lambda = 1$, you get MC

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

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

Forward-view TD(λ)



- 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 Random Walk