Chapter 4. Model-Free Prediction (=Policy Evaluation)

1. Introduction

#. Agent가 Env.에 대한 정보(MDP[1])없이 놓여 졌을 때(=Model-Free)[2], 주어진 Policy를 바탕으로 State Value Function[3]을 구하는 것을 의미하는 Prediction(=Policy Evaluation)의 구체적인 2가지 방법 MC와 TD에 대해 배운다.

MC와 TD를 통해 구한 State Value Function이 수렴하는지를 살펴보고, 특히 TD를 이용하여 Prediction을 할 때 사용하는 Policy가 Optimal이 아니므로 Bellman Expectation Equation을 사용하여 Prediction하는 법을 살펴본다.

a. Prediction에서는 State Value Function만 구하는 것이다. Action-State Value Function은 Control에서 다룬다.

1) Reward와 State Transition에 대한 정보
2) Reward Function과 State Transition Probability가 Unknown인 상황
3) Return의 기댓값을 의미한다. $E[G_t|S_t = s]$

2. Monte-Carlo Learning (MC)

2-1. Monte-Carlo Learning?

#. Terminal State가 존재하는 Episode마다 Agent가 주어진 Policy에 따라 직접 사건을 실행하면서 얻은 Return(G_t)값들을 저장하였다가 평균을 내어 State Value Function($v_\pi(s) = E[G_t|S_t = s]$)을 구하는 방법이다. MC에는 2가지 방법이 존재하는데 Episode마다 State의 방문을 어떻게 기록할 것인지에 따라 나뉜다.

a. Policy는 Stochastic Mapping from State to Probabilities of Actions이므로 매 Episode마다 Return 값이 다르다.

b. Policy Evaluation과 Prediction은 같은 의미이며, 이 방법을 Monte-Carlo Policy Evaluation이라한다. 따라서 Value function의 정의대로 Expected Return을 쓰는 것이 아닌, Empirical Mean을 사용한다.

2-2. 2종류의 MC Learning

a. 중요한 전제 조건은 모든 State를 방문하는 Policy가 주어져있어야 한다는 것이다. 왜냐하면 모든 State를 평가해야하기 때문이다.

2-2-1. First-Visit Monte-Carlo Policy Evaluation

#. 한 Episode내에서 어느 State를 여러 번 방문하더라도 그 State에 대한 방문 횟수를 한번으로 기록하고 여러 번의 Episode후 축적된 Return을 방문횟수로 나누어 State Value를 구하는 방법이다. 대수의 법칙에 의해 Episode 시행 수가 많아질수록 True Value로 수렴한다. (Unbiased)

a. 이를 수식으로 표현하면 아래와 같다.

Increment Counter	$N(s) \leftarrow N(s) + 1$
Increment Cumulated Return	$S(s) \leftarrow S(s) + G_t$
State Value	$V(s) = S(s)/N(s)$
Law of Large Number	$V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

2-2-2. Every-Visit Monte-Carlo Policy Evaluation

#. 한 Episode내에서 어느 State를 매번 방문하더라도 그 State에 대한 방문 횟수를 인정해주는 것이며 나머지는 First-Visit과 동일하다.

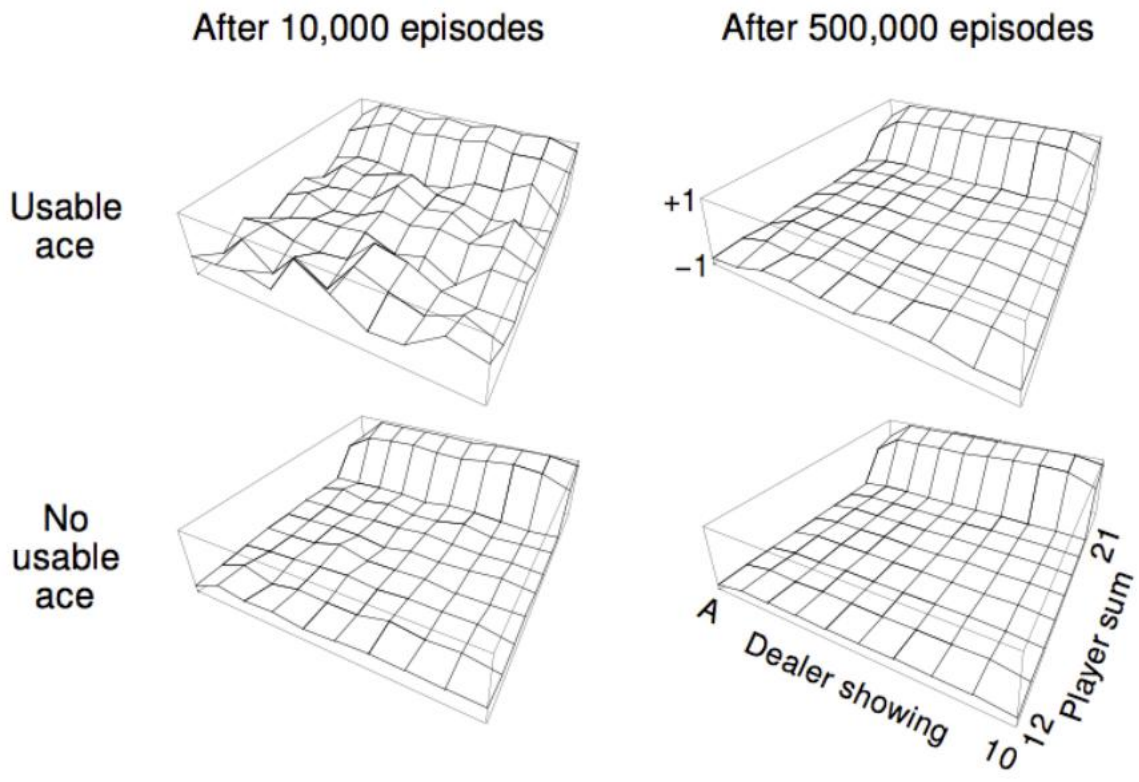
2-3. Blackjack Example in MC

a. Game's Rule : 딜러와 내가 각각 2장씩을 받고, 딜러는 한 장의 카드를 오픈한 상태에서, 두 카드의 합이 21에 더 가까운 사람이 이기고, 21을 넘기는 무조건 진다.

b. Reinforcement Learning Settings : 아래와 같다.

State	• 현재 내 카드의 합 = (12-21) • 딜러가 오픈한 카드 • Useable Ace여부
Action	• Stick : 카드를 안 받음 • Twist : 카드 한 장을 더 받음
Reward for Each Action	• 합이 21이하인 상황에서 +1 : > 딜러의 합 0 : == 딜러의 합 -1 : < 딜러의 합
Transition	• 내 카드의 합이 < 12 이면 자동적으로 Twist Action

c. Result of Simulation : 주어진 Policy는 내 카드의 합이 20이상이면 Stick, 그렇지 않으면 Twist이며, 이를 무작정 따랐을 때, Value Function의 변화 과정은 아래와 같다.



2-4. Incremental Monte-Carlo Updates

#. 앞에서 살펴본 방법인 각 State마다 G_t을 저장하여 한 번에 평균을 계산한 방법($S(s)/N(s)$)은 비효율적이다. 따라서 Incremental Mean개념을 도입하여 Empirical Mean을 Update하는 방식인데, 이전까지 구한 Empirical Mean($v(S_t)$)과 New Observation(G_t)과의 차이를 Error-Term으로 설정하여 이 값을 Step-Size만큼 반영하여 Empirical Mean($v(S_t)$)을 Update하는 방법이다.

a. Incremental Mean :

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}(x_k + \sum_{j=1}^{k-1} x_j) = \frac{1}{k}(x_k + (k-1)\mu_{k-1}) = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

μ_{k-1}	x_k	$x_k - \mu_{k-1}$
Empirical Mean = Old Data	New Observation = New Data	Direction For Update

b. 이를 수식으로 표현하면 아래와 같다.

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

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

위의 식을 해석하면 지금까지 구한 Empirical Mean $V(S_t)$에다 현 Episode 후에 얻은 G_t와 $V(S_t)$의 차이(Error-Term)에 Step-Size만큼 Weight를 준 것을 더한 모양이다.

c. 이러한 방식은 MDP가 조금씩 바뀌는 Non-Stationary 상황에 유용하게 쓰일 수 있다. 왜냐하면 Step-Size를 작게 했을 때, Update에 반영되는 Error-Term의 영향이 줄어드는데 이는 New Observation가 Update에 끼치는 영향이 적다는 의미이다. 이는 학습 속도가 다소 늦지만 Stable한 장점이 있다.

d. New Observation G_t을 구하기 위해선 반드시 Terminal State가 존재해야하는 한계점이 있다.

3. Temporal Difference Learning ($TD(0)$)

3-1. Temporal-Difference Learning?

#. Terminal State가 존재하지 않아도 Incremental Update가 가능하며, 현재 State Value의 Update는 한 Step 만 더 갔을 때 받는 Reward와 더 간 State Value의 합과 현재 State Value의 차이로 Update하는 Model-Free Prediction 기법 중 하나이다.

a. 이를 수식으로 표현하면 아래와 같으며 학습의 목적은 TD-Error값을 줄이는 것이다.

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

$V(S_t)$	Empirical Data (Old Data)
$R_{t+1} + \gamma V(S_{t+1})$	New Observation (New Data)

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

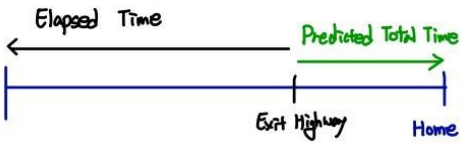
b. MC는 G_t, 즉 한 Episode후에 받는 Return(Discounted Cumulated Sum of Rewards)의 방향으로 Update가 진행되지만, TD는 $R_{t+1} + \gamma V(S_{t+1})$, 즉 한 Step앞의 방향으로 Update가 진행되는 차이가 있다.

c. 그럼에도 MC, TD 모두 대수의 법칙(Law of Large Number)에 의해 수렴한다. (Unbiased)

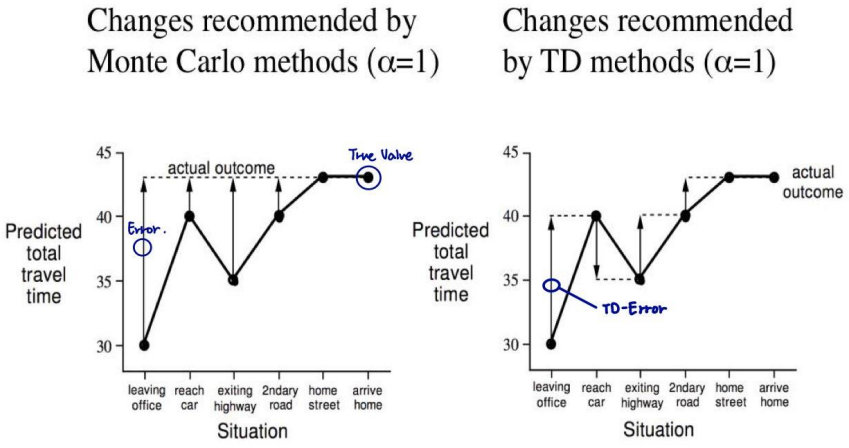
4-1. Driving Home Example : Predict Total Time

a. 아래와 같이 6개의 State가 존재하며 목적은 MC, TD를 사용하여 Office에서부터 Home까지의 Predicted Total Time(=State Value)을 구하는 과정의 차이를 살펴보는 것이다. 이 때, 각 State마다 Home까지의 Predicted Total Time은 변화한다.

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



b. MC의 경우 실제로 Terminal State인 Arrive Home에 오니 43분(True Value)이 걸렸다. 따라서 이전의 State Value(=Predicted Total Time)들을 43분으로 모두 Update해주고, TD의 경우 한 Step 더 갔을 때의 Predicted Total Time(=State Value)으로 Update해주는 것을 알 수 있다.



4-2. 학습가능 시점과 필요한 조건 MC vs. TD(0)

#. TD는 Terminal State가 없어도 다음 State Value를 이용하여 현재 State Value의 Update가 가능하지만 MC는 하나의 Episode가 끝나야지만 현재 State Value의 Update가 가능하다. 이 차이 때문에 TD는 Non-terminal 상황에서 학습이 가능하지만 MC는 Terminal한 상황에서만 학습이 가능하다.

4-3. Bias와 Variance Trade-Off 관점에서의 MC vs. TD(0)

#. 아래의 정리된 표를 참고하면 된다.

	Variance	Bias	Properties
Monte-Carlo	High	0	• 0 Bias로 인해 수렴성이 좋고 이 때문에 Neural Network를 사용하여 Function Approximation으로 표현한 State Value Function도 잘 수렴한다. • 실제 값(Terminal State에 도착하여 구한 True State Value)으로 Update를 진행하기 때문에 초기 값에 민감하지도 않다.
Temporal Difference	Low	Some	• MC보다 사용하기에 효율적이다. • some bias 때문에 수렴가능성에 의문이 있지만, 운 좋게 수렴한다. 특히 $TD(0)$는 v_π에 수렴한다. • Some bias 때문에 function approximation은 항상 수렴하지 않는다. • 추측 치(한 Step 다음의 State Value)로 Update를 진행하기 때문에 초기 값에 민감하다.

a. Variance는 Uncertainty를 의미하고 이는 곧 Random적인 속성이 얼마나 있는지를 측정하는 지표이다. Bias는 True Value에의 수렴 가능성을 표시하는 척도인데, Unbiased는 True Value에 수렴할 가능성이 매우 높다는 것이다.

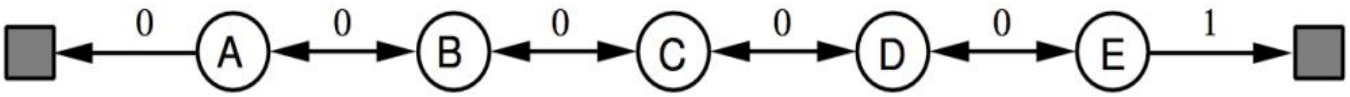
b. MC의 Error-Term ($G_t - V(s)$)에서의 G_t는 여러 State를 주어진 Policy에 따라 확률적인 속성으로 지나면서 하나의 Episode가 끝났을 때의 값이므로 Random property가 굉장히 많이 있다. 따라서 Variance가 굉장히 높다. 하지만 실제 값으로 Update하기 때문에 Bias가 0이다.

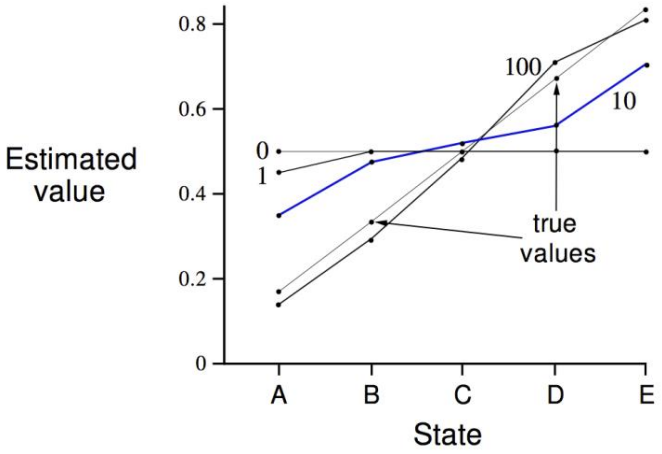
c. TD의 Error-Term ($R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$)에서의 $R_{t+1} + \gamma V(s_{t+1})$는 한 Step 이후의 값이므로 Random Property가 적다. 따라서 Variance가 굉장히 낮다. 하지만 바로 Guess by Guess이므로 수렴가능성이 매우 낮아 Bias가 높다.

d. 결국 각각의 Error-term들의 속성 때문에 MC, TD는 Variance/Bias trade off 관점에서 차이점이 존재한다.

4-4. Random Walk Example

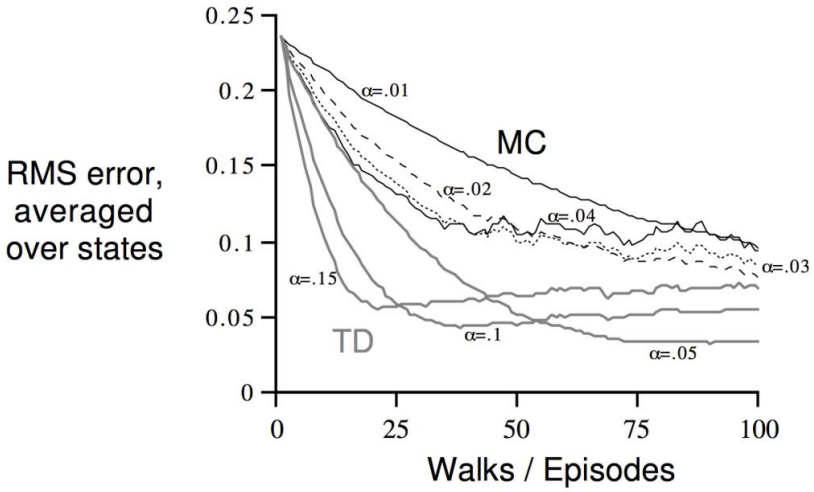
a. 아래의 그림처럼, 6개의 State가 있으며 좌우로 움직이는 Policy가 있다고 하자. TD를 사용하였을 때의 결과는 그래프에 제시되어 있으며, 100번 째 Episode가 지났을 때, True Value for Each State에 가까워지는 것을 알 수 있다.





b. 아래의 그래프는 위의 Random Walk Env.에서 MC와 TD를 각각 α값에 따라 학습시켰을 때의 RMS Error의 변화를 학습 진행에 따라 나타낸 그래프이다. 즉, 위에서처럼 True State Value는 알려져 있기 때문에 MC와 TD가 각각 예상한 State Value와의 차이의 평균값이 RMS error값이다.

c. 전반적으로 RMS Error 값은 감소하지만 Step-Size(α)가 커질수록 다시 증가하기도 하며 전반적으로 TD가 감소하는 속도가 더 빠르다.



4-5. Batch[4] MC and TD(0)

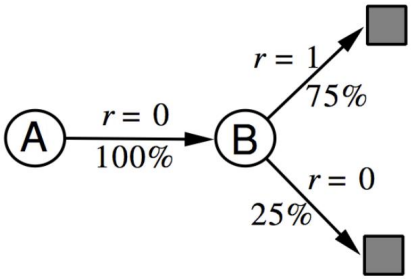
a. 경험하는 Episode가 무한정 늘어나면 MC 또는 TD 방법으로 Estimation한 $V(s)$는 $V_\pi(s)$로 수렴함을 알 수 있다. 하지만 만약 아래의 K개의 제한된 Episode만 있는 상황(T는 time step)에서는 MC와 TD의 차이는 어떨까? 이는 'Certainty Equivalence'의 개념으로 설명할 수 있는데 AB Example을 통해 알아보자.

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

4-5-1. AB Example

a. 아래의 그림과 같이 2개의 State A, B만 존재하고, Discounting이 없으며 Initial State 또한 Random하게 설정된다. $K=8$개의 Episode만 경험하였다. 그리고 Model-Free이므로 State Transition Probability와 Reward는 알려지지 않은 상태에서 $v(A), v(B)$을 MC, TD로 어떻게 Estimation할 수 있을까?

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



4-5-2. Certainty Equivalence

#. 아래의 정리된 표를 참고하자.

	Target	특징	$V(A)$
MC	$\sum_{k=1}^{K}\sum_{t=1}^{T_k}\left(G_t^k - V(s_t^k)\right)^2$	• MSE를 최소화하는 방향으로 수렴함. 따라서 관찰된 return과 맞도록 학습한다.	0
TD(0)	$\hat{\mathcal{P}}_{s,s'}^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$ $\hat{\mathcal{R}}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$	• Max Likelihood Markov Model의 해를 구하는 방향으로 수렴함. 따라서 MDP ($\langle S, A, \hat{P}, \hat{R}, \gamma \rangle$)가 Data와 맞도록 학습한다.	0.75

a. MC의 경우 State A로 갔을 때, 무조건 State B로 간 후 Reward 0을 받기 때문에 State Value A가 0이 된다. 하지만 TD의 경우 이미 State B에서 Reward 1을 받은 경험이 있으므로, A에서 B로 한 Step만 가서 A를 Update하기 때문에 A의 State Value는 0.75가 된다.

4) 전체의 일부만 존재한다는 의미

4-5-3. Markov property 사용 여부관점에서의 MC vs. TD

#. TD는 Markov Property를 사용하여 State Value를 Estimation하는 모델이다. 따라서 Markov Env.에서 효과적으로 사용할 수 있다. 반면에 MC는 이를 사용하지 않고 MSE를 최소화하는 것이 목적인 모델이다. 따라서 Non-Markov Env.에서 효과적으로 쓸 수 있는 차이점이 있다.

4-6. Unified View

4-6-1. Bootstrapping and Sampling

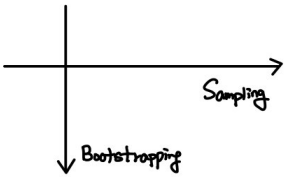
#. Bootstrapping은 Episode를 끝까지 하지 않고 추측치를 이용하여 State Value를 Update하는 방식을 의미하고, Sampling은 Episode를 진행하는 과정에서 가능한 모든 경우를 경험하지 않고 일부만 가지고 State Value를 Update하는 방식을 의미한다.

a. 각 model에 따른 Bootstrapping과 Sampling[5] 여부는 아래와 같다.

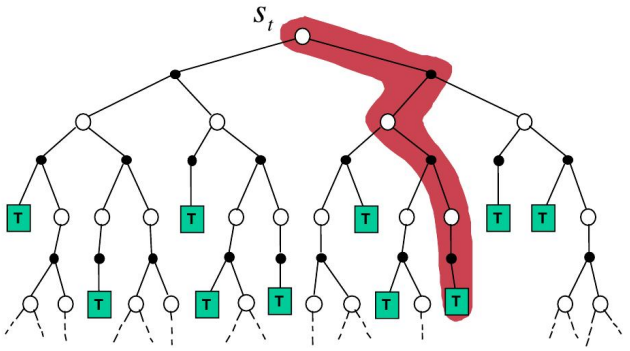
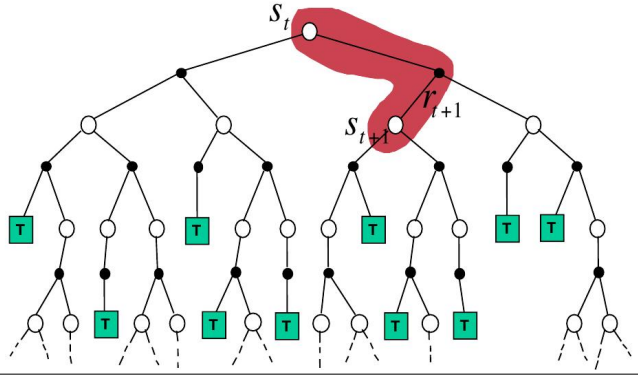
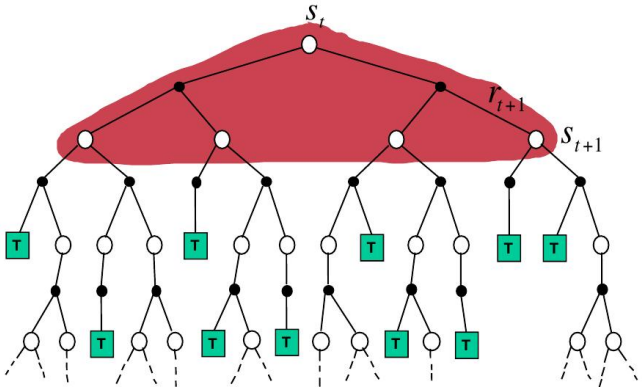
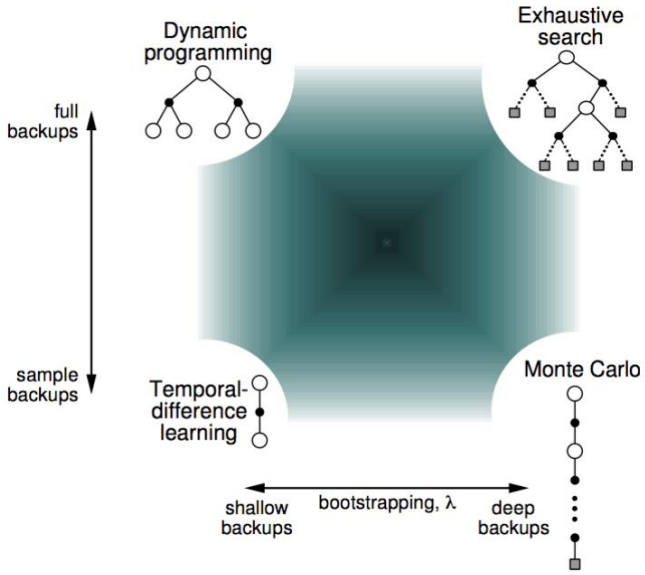
	Bootstrapping	Sampling
MC	x	o
TD	o	o
DP	o	x

5) sampling은 agent가 policy를 따라 움직이며 data를 수집하는 것을 의미한다.

4-6-2. Unified View



#. 아래의 표를 참고하자.

Monte-Carlo Backup	Temporal-Difference Backup
$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$ 	$V(S_t) \leftarrow V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ 
Dynamic Programming Backup	**Unified View of Reinforcement Learning**
$V(S_t) \leftarrow \mathbb{E}_\pi\,[R_{t+1} + \gamma V(S_{t+1})]$ 	

a. 강화학습의 통합적인 View를 보게 되면, 한축은 Bootstrapping, 다른 축은 Sampling이다. MC는 Deep Backup이라 하며, Bootstrapping을 하지 않고 Sampling만 한다. TD는 Shallow Backup이라 하며, Bootstrapping과 Sampling을 한다. DP는 Full Backup이므로 Sampling을 하지 않으며, Bootstrapping을 하게 된다. 마지막으로 Exhaustive Search는 모든 트리에서 가능한 모든 경우를 탐색하므로 Learning이 아니며 비용이 많이 든다.

#. Eligibility Trace의 역할은 TD와 MC를 통합해주고, MC를 Online Learning, Continuous Problem에 적용할 수 있도록 해주는 것이다.

5-1. n-step TD Learning

#. 기존 TD(0)의 경우, 바로 다음 Step에서의 추측치를 이용하여 현 State Value를 Update했지만 n-Step TD에서는 $n-$Step만큼 더 가서 도착한 State의 추측 값과 Discounted Cumulated Rewards를 이용하여 현재 State Value를 Update하는 방법이다.

a. n-step만큼 가서 얻을 수 있는 Return은 아래와 같다. 즉 n-step만큼 Reward를 받고 S_{t+n}에서는 추측치를 더해주는 것이다.

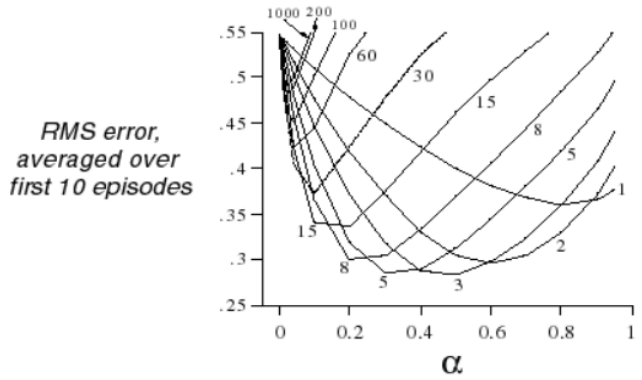
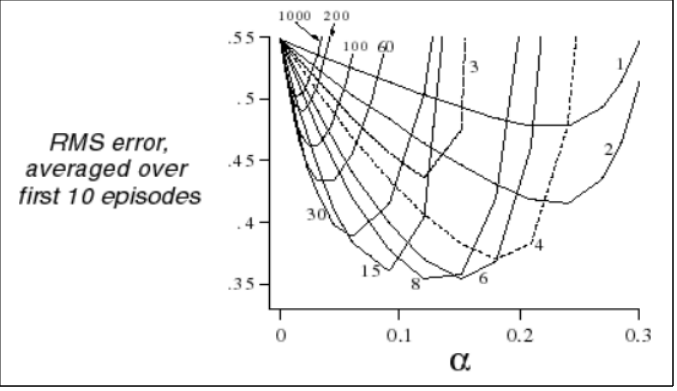
$$\begin{aligned} n=1 \quad (TD) \quad & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\ n=2 \quad & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\ \vdots \quad & \vdots \\ n=\infty \quad (MC) \quad & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T \end{aligned}$$

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

b. 따라서 n-step TD의 Update식은 아래와 같다.

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

c. 아래는 n-step에 따른 Random Walk Example인데, Episode가 10개 밖에 되지 않으므로 Return으로 Update 하는 MC는 성능이 떨어진다. 왜냐하면 MC의 경우, Update의 횟수가 10회 밖에 되질 않기 때문이다.

ON-Line[6] n-step TD	OFF-Line[7] n-step TD
	

d. 위의 그래프에서처럼 TD(0)와 MC사이에 Sweet Spot이 존재한다.

6) Episode를 진행하면서 바로 State Value를 Update하는 방식.
7) Terminal State까지 진행한 후 해당 State Value를 Update하는 방식.

5-2. $\lambda-$Return[8] : Forward View TD(λ)[9]

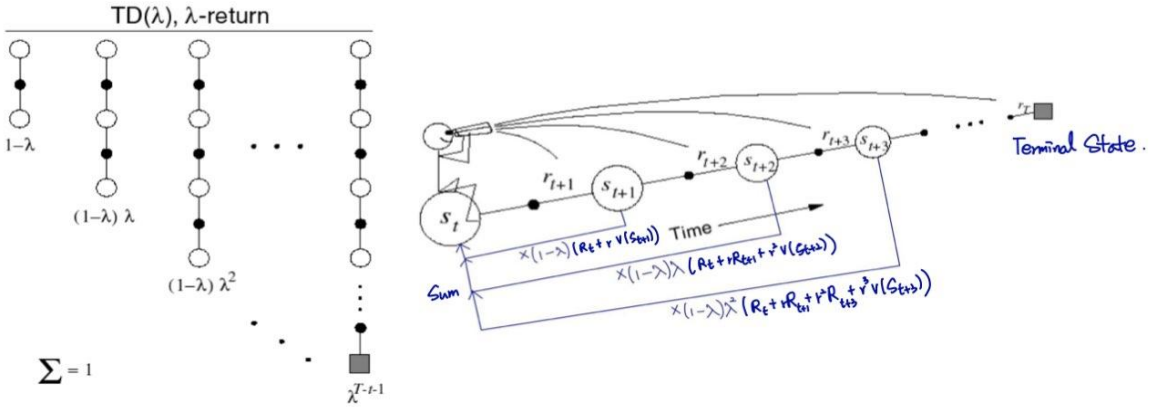
#. S_t에서 출발하여 Terminal State까지 가는 과정에서, 매 Step마다 얻은 Return[10]들을 각각 $(1-\lambda)\lambda^{n-1}$만큼 가중 평균[11]하여 S_t의 Value를 Update하는 방법이다. 따라서 MC와 같이 Terminal State가 필요하며, TD(0)의 장점이 사라지는 단점이 있다.

Update식은 아래와 같으며 각 Step에서의 Return에 가중치 λ^{n-1}을 곱해주는 방식이다. 하지만 단점은 구현하기가 어렵다는 것이다. 왜냐하면 현재 t시점의 State를 Update하기 위해선 t시점 이후에 나오는 것을 이용해야 하기 때문이다. 따라서 실제적으로는 Forward View와 Update가 거의 동일하고 최근 들어 강력한 성능을 내는 Backward View(TD(λ))를 사용한다.

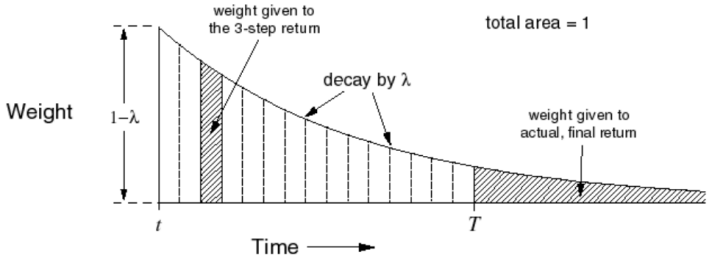
$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)} \quad (0 \le \lambda \le 1)$$

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

$(1-\lambda)$	Sum=1을 위한 Normalization Term	λ^{n-1}	Return Weight



a. 아래의 그림은 λ가 시간에 따라 변화하는 양상을 그림으로 표현한 것이다. 이렇듯 기하 평균을 사용하는 이유는 Memory Less하게 학습하여 계산상의 효율성이 있기 때문이다. 즉 중간에 값을 저장하지 않기 때문에 TD(0)와 같은 계산양이 필요하다.



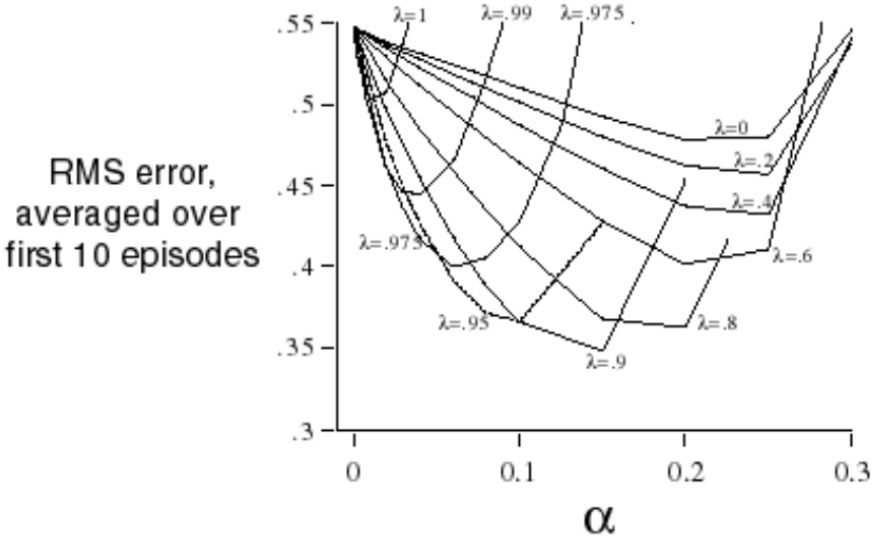
8) 여기에서 λ의 의미는 Return G_t의 Weight의 의미이다.

9) Episode가 끝나야 update가 끝나므로 OFF-LINE Learning이다.

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

11) 가중 평균을 하는 이유는 아래의 그림과 같이 완만하게 깎이도록 하기 위함이다.

b. Large Random Walk Example에 적용하였을 때, 그 결과는 아래와 같다. 아래의 그래프에서와 같이 Forward View TD(λ)에서는 각 Step에서의 Return($G_t = R_t + \gamma R_{t+1} + ... + \gamma^n V(S_{t+n})$)에 대한 Weight 값인 λ을 정해주는 것이 중요하다.



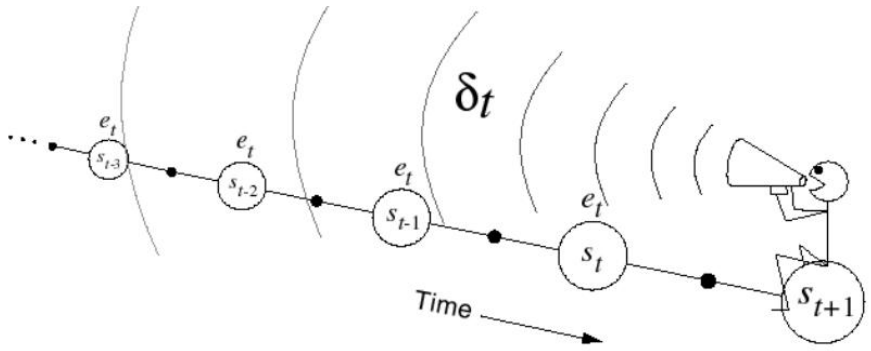
5-3. TD(λ)[12] : Backward View TD(λ)[13]

#. 각 State 마다 (Action-State Value의 경우 Pair(s, a)에 Eligibility Trace 값이 저장된다.) Frequency, Recency Heuristic을 혼합한 Eligibility Traces 값을 기록한 후, 이 값이 큰 State를 더 많이 Update해주는 방식으로 아래와 같이 TD(0) Update식 중 Error-Term에 Eligibility Trace 값을 곱해주어 진행한다. 즉 TD-Error 값에 Weight로써 Update를 조절해주는 역할을 하는 것이다. (Linear Function Approximation을 사용하여 Value Function을 표현한 경우 Eligibility Trace는 Estimation Value를 결정하는 Weight Vector의 학습에 영향을 준다.)

<table>
<tr><th colspan="2">Eligibility Traces</th><th>Backward view TD update</th></tr>
<tr><td colspan="2">$E_0(s) = 0$

$E_t(s) = \gamma\lambda E_{t-1}(s) + 1(S_t = s)$</td><td rowspan="3">$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

$V(S) \leftarrow V(S) + \alpha\delta_t E_t(S)$</td></tr>
<tr><td>Recency</td><td>$\gamma\lambda E_{t-1}(s)$</td></tr>
<tr><td>Frequency</td><td>$1(S_t = s)$</td></tr>
</table>



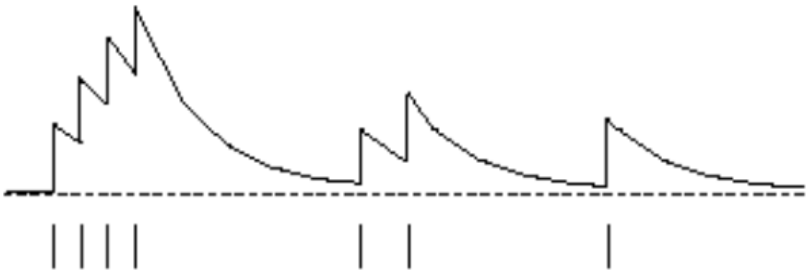
a. 이 Eligibility Trace 값은 매 Episode마다 0으로 초기화 해주어야 한다.

b. Agent가 지나왔던 State(or Action-State Pair)들을 이용하여 학습하므로 Non-Terminal Env.에서 학습이 가능하다.

c. 각 state마다 Eligibility traces 값을 가지고 있으며 주로 Tabular Method를 사용하여 구현한다. Update는 매 Time-Step 마다 Eligibility Trace Table에 $\gamma\lambda$을 곱해주고 방문여부($1 \text{ or } 0$) 값을 더해준다. 그렇게 되면 State를 방문 할 때 마다 1을 올려주고, 방문하지 않을 때는 $\gamma\lambda$만큼 지속적인 Decay가 된다. 그 과정은 아래의 그림으로 표현되어 있다. 따라서 최근, 많이 방문할수록 이 값은 자연스레 커지게 되고(책임소재가 커진다.) 더 큰 Update를 할 수 있다.

d. 아래의 그림은 특정 State s의 Eligibility Trace 값의 변화 양상을 보여준다.

12) 여기에서 λ의 의미는 Eligibility Trace를 사용한다는 의미이며, Decay Scalar for Eligibility Trace의 역할을 한다.
13) Episode가 끝나지 않아도 매 step마다 update가 가능하므로 ON-LINE Learning이다. 그리고 Terminal state가 없어도 학습이 가능하다.



e. 수학적으로 Forward TD(λ)와 동일한 의미를 가지며, 과거의 책임소재를 기록하면서 진행하기 때문에 매 Step마다 Update(ON-LINE)가 가능하다. 결국 TD(0)의 장점과 Forward TD(λ)의 장점을 모두 가지고 있다.

5-4. Relationship Between Forward and Backward TD

#. 아래의 표를 참고하자.

	TD(λ) vs. TD(0)	**TD(λ) vs. MC**
조건	$\lambda = 0$이면 Current State만 Update 되므로	$\lambda = 1$ 이고 OFF-LINE이면
관계	$E_t(s) = 1(S_t = s)$ $V(s) \leftarrow V(s) + \alpha\delta_t E_t(s)$ $V(s_t) \leftarrow V(s_t) + \alpha\delta_t$ $\therefore TD(\lambda) = TD(0)$	1. Forward View TD(1) = MC 이다. 2. Update의 합은 Forward-View와 Backward-View가 동일하다. $\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \sum_{t=1}^{T} \alpha\left(G_t^{\lambda} - V(S_t)\right) 1(S_t = s)$