

Dynamic programming

07 Dynamic
24/01/04 (32)
27/163
20/163

Substructure: 복잡한 문제를 분할하여 계산

Table structure: 계산된 것들을 테이블에 저장하여 반복 사용

Bottom UP Computation: 테이블을 사용해서 조금 더 큰 문제 해답 \rightarrow 큰 문제

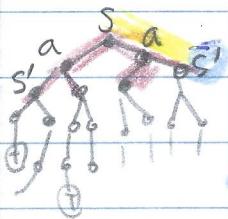
DP에 좋은 문제

Optimal substructure: 분할하여 푸는게 좋은 문제

Overlapping sub-problem: 작은 문제들의 해답을 반복 사용한 문제

예) Markov Decision Process

- Bellman equation $V_{\pi}(s)$ 를 구하기 위해 전 $k-1$ 까지 구해야 함



Dynamic Programming (DP): full Back UP

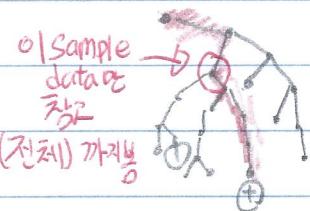
$V(s_t) \leftarrow \mathbb{E}_{\pi} [R_{t+1} + \gamma V(s_{t+1})]$ Bellman expectation 이것을 사용 state-value function $V(s_t)$ 값 갱신

s, s_{t+1} 과 같은 모든 state value function 값을 알아야 expectation \mathbb{E}_{π} 계산 가능
transition probability p 를 알고 있기에 직접 \mathbb{E}_{π} 계산 가능

RL

Monte Carlo (MC): sample multi-step backup

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



Transition probability를 알고
Bellman expectation
sample data
계산

Temporal Difference (TD): Sample backup

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



Monte Carlo
target(기대) 사용 A episode 끝날 때까지.

Temporal Difference
 R_{t+1} 과 s_{t+1} 사용

현재 state s 가 있을 때 Policy π 를 따라 모든 return G_t 의 expectation

Bellman expectation equation: $V_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi}[R_{t+1} + \gamma V_{\pi}(s_{t+1}) | S_t = s]$

Bellman에 따라, return 을 다 계산하기 않아도 immediate reward R_{t+1} 과 next state s_{t+1} 에 대한 V_{π} 를 가지고 계산 가능

Back up: future state의 values 값을 사용해서 현재 state의 value 값을 업데이트

Generalized policy Iteration (GPI) 중요

Policy Iteration 아래 두개를 고대로 사용

기법

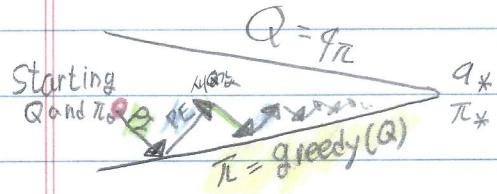
Policy Evaluation: Current policy가 있을 때, policy에서 state-value 값을 도달 할 때까지 한다.

Policy Improvement: Current value function을 기반으로 greedy를 사용 policy를 업데이트

Completes 할 때까지 (Policy를 정한 뒤 Value function 업데이트시 full back 사용 도달 까지. (필요 이상))

GPI

Sample backup에 대해서만 data update. Current policy에 대해서 정확한 true value 값을 직접 찾을 수는 없고 approximated (근접) Value function을 찾



Q Value table을 바탕으로 policy Improvement 함
그때는 greedy policy 사용. 새로운 policy를 가지고

Policy Evaluation 함, 그러나 value function이
도달 할 때까지 갈 수 없음. Samples data만 가지고 하기 때문 (완벽한 정보 x, 일부만 z)
그래서 approximated 되어 있는 value function을 계산하게 되는, 그래서 중간에 멈춤 ↗

GPI에서 PE, PI stabilize (번복 없음) 이음.

~~$Q^* = \pi^*$~~

결과로 나온 Q-function은 Bellman optimality equation을 만족함
Value function은 policy가 optimal이란 뜻.

결론 DP에서는 policy Evaluation, policy Improvement를 하는데, 고정에서 transition probability를 알고 있어 도달 할 때까지 policy Evaluation을 계속 업데이트 했음.

GPI는 Samples data이기 때문에 도달 할 때까지 할 수 있음 approximated를 사용하여 함
그래서 PE 가다가 중간에 멈춤. PE, PI가 stabilize한 순간이 음, 그때 얻어지는
Value function, policy는 optimal임.

DL과 RL의 차이

DP, RL은 tabular updating method.

1. DP는 각 iteration(번복) 업데이트 때 테이블의 모든 값을 사용 (full backup)

$V(s)$ 를 사용 ($Q(s, a) \times$) 이유는 $|S| \ll |\{s, a\}|$

\Rightarrow Greedy policy improvement over $V(s)$ requires known MDP.

$$\pi'(s) = \arg \max_a \sum_{s', r} \rho(s', r | s, a) [r + \gamma V^\pi(s')]$$

value function update 공식

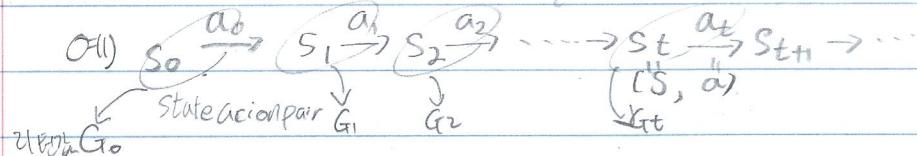
$V(s)$	Cool	Warm	over
V_2	3.0	2.5	0
V_1	2	1	0
V_0	0	0	0

optimal policy를 값最大化시키는 a 를 찾은 후 state s 에서의 액션으로 결정

$\pi'(s)$ 는 deterministic. 테이블의 $V^\pi(s)$ 값이 기록되어 있음.

$Q(s', r | s, a)$ 알다가 Value function update 공식 계산이 가능 (known MDP를 가정)

2. RL 각 iteration 때 Sample data를 사용.



리워드 값으로 Q테이블 업데이트

Q_k	\uparrow	\downarrow	\leftarrow	\Rightarrow	\leftarrow action	Q_k	\uparrow	\downarrow	\leftarrow	\Rightarrow
1	0.40	0.41	0.42	0.43		1	0.40	0.41	G_0 0.44	0.43
2	0.43	0.42	0.41	0.40	G_1 업데이트 \Rightarrow	2	0.43	G_1 0.46	G_2 0.48	0.40
3	:	:	:	:		3	:			
⋮	:	:	:	:		⋮				
9						9				

↑ State s

\Rightarrow Greedy policy improvement over $Q(s, a)$ can work for model free

$$\pi'(s) = \arg \max_a Q^\pi(s, a)$$

* RL breaks the curse of dimensionality of full back

모든 state을 다 고려하면 낭비

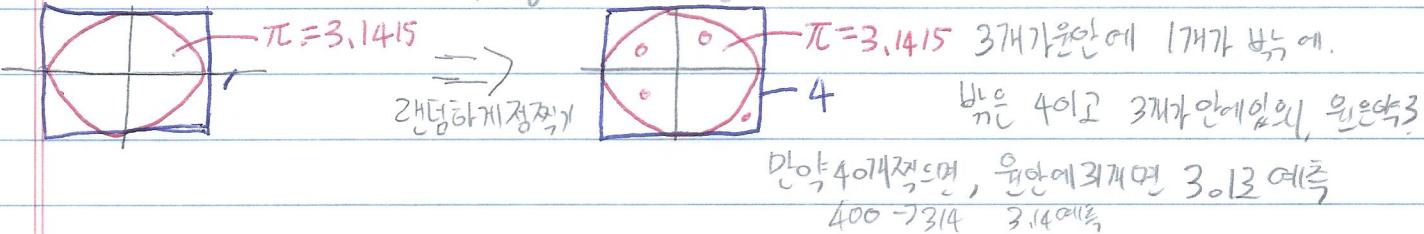
in DP by using sample back up of real experience

sample만 가지고 흐름을 찾기 가능

Monte Carlo method (MC) 개념

Monte Carlo의 Repeated random sampling 사용해서 특정한 numerical (수치) results 얻어내는 Computations(계산) relied (바탕에)

예) PI 값을 random sampling으로 구하기를 염두한다면?



MC

1. tabular updating (RL : action-value function table Q-table)

2. Model-free

* MC의 특징

Episode-by-episode : 1. Sample을 얻어진행, | Episode-by-episode GPI : PE와 PI를 각 에피소드마다 반복

- Episodic task : 시간의 지남에 유호한 시간내에 끝나는 task

- Continuous task : 끝이 없이 진행되는 task

MC는 한 게임이 끝날 때 상태에서 return을 가지고 진행을 하기 때문에, episodic task에 맞다.

일정 시간이 지나, 일이 마치면, discount total reward인 return값을 얻음

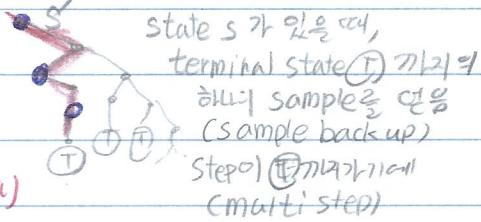
MC Policy Iteration은 GPI(Generalized Policy Iteration)을 기본으로 하고, Episode-by-episode로

Sample을 얻어 진행 (Sample multi-step backup)

PE은 $Q(s, a)$ 를 Evaluation함. True Q function $q_{\pi}(s, a)$ 를

estimating 할 때까지 진행 - PE estimating $Q(s, a) = q_{\pi}(s, a)$

그리고 PI는 ϵ -greedy PI를 사용



MC는

Sample로 주어진 전체 trajectories (terminal state까지)로 학습을 함, 매번 주어진 Single episode를 이용하여 update를 함

Sampled episodes (샘플로 주어진 에피소드)마다, G_t (리턴값)을 얻음, G_t 의 expectation이 Q-value 계산에 사용. 그리고 MC는 Average를 returns하여 Value function 값을 정함

Value = Average of returns G_t of sampled episodes

Monte Carlo method 개념

MC 장점 : 1. State Space에서 필요로 하는 state는 small subset뿐임. 예) 비독의 경우(state)은 (기회 학습 도함) 무척 많지만, sample로 나오는 action state pair만 있음, 대부분 무시(실제 일어나지 않을 때) 흐름적

2. Markov property가 깨져도 크게 영향 받지 않음

Bootstrapping : Q-Value 업데이트 시 return G_t 를 사용하는 방법과 immediate reward와 next state에서

Q-Value를 사용하는 방법이 있음 ($R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$)

(근사치)

true value G_t , state-action pair의 실제 true Q값에 대한 estimates. $Q(S_{t+1}, A_{t+1})$

즉, $Q(S_{t+1}, A_{t+1})$ or $Q(S_{t+n}, A_{t+n})$ 을 사용하여 estimates하면 Bootstrapping.

Monte Carlo는 true value G_t 를 사용해 update - no bootstrapping

장점 2번 설명 : Markov property는 과거는 무시, 현재, $t+1$ (미래)를 가지고 값을 도출하는 : $Q(S_{t+1}, A_{t+1})$ 사용에서 잘 적용됨. 그러나 MC는 G_t 를 사용하므로, 게임이 끝날 때까지 진행. 현재 state에서 Q값 업데이트시 Next state($t+1$) 뿐 아니라 다음 states도 전부 적용($t+n$), 얻어지는 reward 값을 total sum of return을 사용함. 즉, 현재 state의 약간 오차가 시장으로 지양 끝에 있는 결과 사용하기에 크게 영향 받지 않음. 예시 : Value function 값을 estimate 시 next state의 value-estimate를 기본으로 하지 않음.

MC prediction (Policy Evaluation)

24/11/07-28

4:30 ~ 50

45/170

3:40 ~ 50

목적 :

현재까지 얻어진 current policy π 기준에 실제 경험한 entire episode로 True Q-value (q_{π})을 험하여 가치값을 도출.

하나의 에피소드에 전체 trajectory : $S_0, A_0, R_1, S_1, \dots, R_T$ (Terminal step) + MC episode task (반드시 같은 개수)

t-step

return : $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1} R_T$. 설명: episode trajectory ... S_t, A_t, \dots, R_t
return 값을 토대로 Q-function update \rightarrow L_t 리워드 = $R_{t+1} + \dots$

action-value function : $q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$

설명: true q_{π} 값은 each state, action pair에 대해서 state와 Action이 present일 때 present state에서 얻어지는 return 등의 expectation을 계산한 값 = 실제 $q_{\pi}(s, a)$

하지만 MC는 Samples를 사용으로 expectation E 를 계산할 수 있음

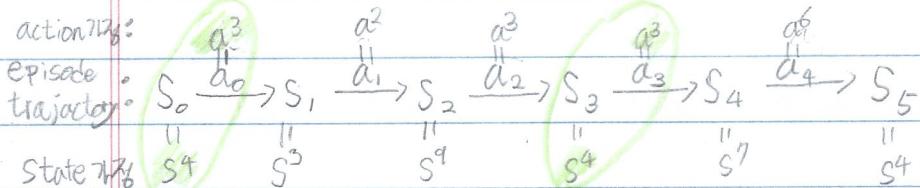
empirical mean return : 그래서 expected return 대신 empirical mean return을 MC policy Evaluation 사용.

mean return : $\frac{1}{n} \sum G_t$ 설명: episode \rightarrow samples \rightarrow return을 계속 합을 G_t . n 개를 얻어 sum을 한 뒤 $\frac{1}{n}$ 으로 나누면 제작

그것 state action pair (s, a) 마다 찾음. 2가지 방법 1. First Visit 2. Every visit

First/Every visited 예 : ?

$$S = \{s^1, \dots, s^m\} \quad A = \{a^1, \dots, a^n\}$$



$G_0 (s^4, a^3) G_3$ G_0 와 G_3 의 return 값 얻었고, G_0 만 쓰고 G_3 버리면 First.

여기 episode가 있는으면 First-Every visit 모두 return 값이 누적됨. G_0, G_3 둘 다 쓰면 every

그러면 mean return을 통해 평균을 내기로 함. 평균을 내기 위해 state-action pair가 몇 번

나오는지 카운트 해야 함 \rightarrow increment count.

- increment count : $n(s, a) \leftarrow n(s, a) + 1$ (for all episode experienced) state-action pair count

설명: state s 를 방문해 action a 를 취한 state-action pair가 몇 번 일어났는지 count

- increment total return : $S(s, a) \leftarrow S(s, a) + G_t$ 리턴값들을 sum

설명: 새로운 리턴 G_t 가 얻어질 때마다. 값을 누적

- Value of (s, a) is estimated by mean return : $Q(s, a) = \frac{S(s, a)}{n(s, a)}$ 평균구하기

설명: 리턴값 $S(s, a)$ 를 더한 후에 전체 개수 $n(s, a)$ 로 나누면 mean return (평균리턴) $Q(s, a)$

$Q(s, a) \rightarrow q_{\pi}(s, a)$ as $n(s, a) \rightarrow \infty$ by the law of large numbers (with i.i.d. 독립항등분포, 고려상상) return assumed.

설명: returns 등 i.i.d. 를 만들고 episode $n(s, a) \rightarrow \infty$ 늘려서 (state-action pair도 많아지면) 그들의 평균이 나와서 $Q(s, a)$ 에 도달.

24/01/08

10:15~11:00 46m
11:20~12:20G_t

PE 성질

Empirical mean return을 단순화한 Incremental mean

$$Q \rightarrow \mathbb{E}[G_t - Q(S_t, A_t)]$$

Incremental mean : Partial mean μ_k of a sequence x_1, x_2, \dots is computed incrementally.설명 : 일반적인 sequence x_1, x_2, \dots 가 있을 때 $\frac{1}{k} \sum_{i=1}^k x_i$ --- Partial mean μ_k 이라 함.N_k를 incrementally(점진적) 계산하는 법

$$\mu_k = \frac{1}{k} \sum_{i=1}^k x_i = \frac{1}{k} \left(\sum_{i=1}^{k-1} x_i + x_k \right) = \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})$$

설명 : μ_k 는 $x_1 \sim x_{k-1}$ 까지 값을 sum한 후 k 로 나눈 마지막 k 를 따로 분리. $\sum_{i=1}^{k-1} x_i$ 는 $(k-1)$ 로 나눈면 μ_{k-1} , $\frac{1}{k} \sum_{i=1}^{k-1} x_i$ 는 μ_{k-1}

$$\sum_{i=1}^{k-1} x_i = (k-1)\mu_{k-1}$$
 식을 풀면 $\mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})$ 이 나온다. 그래서 μ_k 는 바로 전까지 계산해온 μ_{k-1} 과 $\frac{1}{k} (x_k - \mu_{k-1})$ 만 더해주면 됨. 또 다른 변수를 가지고 있을 필요가 없음. 전체 sum한 수 없이 μ_k 를 incrementally 계산 가능.
Incremental Monte Carlo Updates : $\mu_k = \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})$ 을 사용하여 $Q(S_t, A_t)$ 을 업데이트 함.Update $Q(s, a)$ incrementally after one episode $S_0, A_0, R_1, S_1, \dots, R_T$.설명 : 하나의 에피소드 $S_0, A_0, R_1, S_1, \dots, R_T$ 가 끝난 때마다 incrementally 하게 Q-Value estimate $Q(s, a)$ 를 업데이트 함.For each state-action pair (S_t, A_t) with return G_t , $\frac{1}{n(S_t, A_t)} [G_t - Q(S_t, A_t)]$: state-action pair (S_t, A_t) 가 주어졌을 때 return 값은 G_t 라고 함.

$$n(S_t, A_t) \leftarrow n(S_t, A_t) + 1 \quad \text{: State-action pair } (S_t, A_t) \text{에 대해서 } n(S_t, A_t) \text{ 를 방문한 횟수를 Count하는 N값이 필요}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{n(S_t, A_t)} [G_t - Q(S_t, A_t)]$$

설명 : $Q(S_t, A_t)$ 는 return G_t 의 때값. $\frac{1}{n(S_t, A_t)} [G_t - Q(S_t, A_t)]$ 에 $\mu_k = \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})$ 을 사용.가장 최근 G_t 에 이전까지 mean return $Q(S_t, A_t)$ 을 배준 후. 횟수 $\frac{1}{n(S_t, A_t)}$ 을 나눠주어서 값을 업데이트.실제 mean return $Q(S_t, A_t)$ 가 됨. $\frac{1}{n(S_t, A_t)}$ 은 시간이 지남수록 점점 많아짐. 시간이 점점 지남수록 최근에 얻은 리턴값 G_t 가 Q-Value estimate $Q(S_t, A_t)$ 에 미치는 영향이 점점 줄어듬. 손해입니다. 시간이 지남수록 좋은 리턴 G_t 가 나오는데. 별 사용하지 않는다면.Constant- α MC Policy Evaluation : 위 방식의 시간이 지남수록 좋은 리턴을 잘 활용하기 위한 방법. $\frac{1}{n(S_t, A_t)}$ 을 상수로 바꿔줌.We practically use a 'step size' α . 예: $\alpha = \frac{1}{10000}$ 로 고정. 실제 mean return $Q(S_t, A_t)$ 와 나를.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [G_t - Q(S_t, A_t)] \xrightarrow{\text{변형}} (1-\alpha)Q + \alpha G_t$$

장점 : As learning progresses, it gets smarter, So recent samples are more important than old ones.

최근에 얻은 리턴 G_t 가 α 를 통한 일정 비율로 참여, $\frac{1}{n(S_t, A_t)}$ 처럼 작아지지 않음. 최근 정보에 영향력 강화

그래서 Non-Stationary problem임 : reward probabilities vary(달라지기) at times, So non-i.i.d)

설명 : Agent의 시간이 지남수록 policy 성능이 향상, reward probabilities가 바뀔 수 있음. i.i.d. 만족, i.d. (Identically distribution),

확률 분포)가 바뀜 (예: 주사위던지기에서 주사위가 좋아짐) α 로 상수값을 사용하는 것이 조금 더 편리2번째 장점: Old episodes are exponentially forgotten because of $(1-\alpha)Q(S_t, A_t)$ term.설명: $(1-\alpha)Q = (1-\alpha)Q(S_t, A_t)$ term, 바로 전 Q-Value estimate $Q(S_t, A_t)$ 이 $(1-\alpha)$ 를 곱함. 1보다 작은 값을 곱하므로 Old episode 를 exponentially하게 감소시킨다. 반복 시 오래된 에피소드 영향력 감소

Monte Carlo Method - PI (E-greedy)

E-greedy 개념

GPI (Generalized Policy Iteration)

주제:

MC Policy Iteration은 DP Policy Iteration을 RL로 적용(samples)으로 GPI를 기반으로 함
 MC가 다른 RL과의 중요한 차이는 episode-by-episode 방식, Terminate 토록할 때까지 episode 전체를 기록하고 하다가样品을 사용. 하나의 sample은 episode가 주어지면 policy iteration 안에 있는 PE(Policy Evaluation)과 PI(Policy Improvement)를 한 번씩 진행. PE는 PE estimating $Q(s, a) = q_{\pi}(s, a)$ and E-greedy PI.

E-greedy policy: To trade-off Exploitation and Exploration, use E-greedy policy. 즉, E는 탐험, D는 탐색

Exploitation (탐색) : 알고 있는 정보에 최선의 결정 예) 막대한 수익

Exploration (탐험) : 알고 있는 정보의 새로운 결정 예) 새로운 속도, 막대한 강기적용

Exploitation by Selecting the action with the highest Q-value while Sampling new episodes, we can refine our policy efficiently from an already promising region in the state-action space.

설명: 가장 큰 Q-value를 가지고 actions를 선택 (greedy policy와 같은 개념, 가능한 Q-value를 갖는 액션 선택.)

언제나 actions(policy)를 가지고 Sampling을 함. State-action Space에서 경험했던 것 중 가장 높은 Q-value를 갖는 선택된 지역에서 best action을 취하기에 효율적으로 policy를 업데이트하게 됨.

그러나 Exploitation (greedy policy)만 사용하면 Sample에서 반복적으로 나온 state-action pair에 대해서는 계속 updated 되기 때문, 그렇지 않으면 new state-action pair는 전혀 사용되지 않아 AI pair의 Q-value가 update

Exploration by Selecting an extra random action with E-probability while Sampling new episodes, we can find a new and maybe more promising region within the action space

설명: Exploration의 highest Q-value selecting the action or 1-E의 확률로 선택 예) $E = 0.0001$, $1-E = 99.99\%$ 는 highest Q-value 갖는 action. $E = 0.0001$, 0.001% 만큼 random action 선택

가보지 않았던 state-action pair에 대해 가능 선택. 새로운 episode를 Sampling 한다.

은 만큼 선택할 수 있으나, 더 좋은 결과를 얻을 수 있다.

기본 아이디어:

Gather enough information (가보지 않았던 new state-action pairs)를 찾고 함 to make the best overall decision (궁극적으로 더 좋은 선택을 할 수도 있다.)

Since the best long-term strategy(전략) may involve(포함) short-term sacrifices(설계)

MC Control (E-Greedy Policy Improvement)

12 MC 2

24/01/09

12:05 21:10 48/110

- Choose the greedy action with probability $1-\epsilon$ and a random action with probability ϵ .

random action \rightarrow
선택 확률 ϵ

$(\frac{\epsilon}{m} \text{ for each of all } m \text{ actions}) \Rightarrow \text{Stochastic policy}$

설명: Policy를 선택 A probability $1-\epsilon$ 확률의 greedy action(highest Q-value) and probability ϵ (ϵ 확률) 만큼은 random action을 선택.

각 행동 액션 선택 \rightarrow 모든 액션의 개수가 총 m 개라고 가정, 각 action마다 $\frac{\epsilon}{m}$ 확률을 적용함. 모든 action에 ϵ 만큼 랜덤하게 적용.

deterministic policy
가장 확률 action은, 무작위로

Stochastic policy
기준은 ϵ 은 랜덤으로 들어감.

* In full backup DP, Policy Improvement uses $\pi'(s) = \arg \max_a Q^\pi(s, a) \Rightarrow \text{deterministic policy}$

설명: \Rightarrow Point full backup을 사용할 때, highest Q-value $Q^\pi(s, a)$ 을 갖는 action $\max_a \pi(s, a)$ 가 state s 의 액션 선택 \Rightarrow deterministic policy.

그리고 지금은 Stochastic으로 바꿔면으로, $\pi(a|s)$ 로 바꾸어야 함.

MC control ②
So all actions are selected with non-zero probability for ensuring continual exploration.

$\pi'(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{m} & \text{if } a = \arg \max_{a'} Q^\pi(s, a') \\ \frac{\epsilon}{m} & \text{Otherwise (m-1 actions)} \end{cases} \leftarrow \text{stochastic policy}$

설명: states에서 각 action마다 확률이 주어짐. 확률이 주어지는 방법: $a = \arg \max_{a'} Q^\pi(s, a')$ highest Q-value
을 갖는 action임. $1-\epsilon$ 의 확률을 제거하고 m 개의 모든 action에 대해서 $\frac{\epsilon}{m}$ 을 추가로 더함. 나머지 $m-1$ 개는 ϵ .

모든 action에 대해 확률이 0이 갖는 압축. All actions are selected with non-zero. 연속적으로 exploration이 보장됨.
근데 왜 $m-1$ actions 만 ϵ 을 추가하지 않는가? $\frac{\epsilon}{m-1}$ 이 $m-1$ actions로 1개는 $1-\epsilon$ 을 할 수 있으니, 그게 보장 때문.

③ Policy Improvement가 전보다 낫다는 것에 보장 되어야 함 예) DP PI $V_{\pi'}(s) \geq V_{\pi}(s)$

보장 ④ For any ϵ -greedy policy π , ϵ -greedy policy π' w.r.t. q_π is always improved.
 $V_{\pi'}(s) \geq V_{\pi}(s)$. 설명: policy π 는 ϵ -greedy policy π' 의 policy에 의해 update된 q_π 의 일부.
이것에 대해 ϵ -greedy를 적용한 policy π' 가 있음. π' 는 항상 improve 됩니다. $V_{\pi'}(s) \geq V_{\pi}(s)$ 보장됨.

MC Control (ϵ -Greedy policy Improvement)

$$\pi'(a|s) = \begin{cases} 1-\epsilon + \frac{\epsilon}{m} & \text{if } a = \arg \max_{a'} Q^\pi(s, a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$$

22:20
3:20

$$\therefore q_\pi(s, \pi'(s)) = \sum_a \pi'(a|s) q_\pi(s, a) = \frac{\epsilon}{m} \sum_a q_\pi(s, a) + (1-\epsilon) \max_a q_\pi(s, a)$$

$$\geq \frac{\epsilon}{m} \sum_a q_\pi(s, a) + (1-\epsilon) \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s, a) \quad (\because \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} = 1)$$

$$= \sum_a \pi(a|s) q_\pi(s, a) = V_\pi(s)$$

$$\text{설명: } - q_\pi(s, \pi'(s)) = \sum_a \pi'(a|s) q_\pi(s, a)$$

↑
현재
↑
이전은
 $\pi'(a|s) = \begin{cases} 1-\epsilon + \frac{\epsilon}{m} & \text{를 적용해야 함.} \\ \frac{\epsilon}{m} & \end{cases}$

$$- \sum_a \pi'(a|s) q_\pi(s, a) = \frac{\epsilon}{m} \sum_a q_\pi(s, a) + (1-\epsilon) \max_a q_\pi(s, a)$$

action a 를 선택하는 데 확률을 따름, $\pi'(a|s) = \begin{cases} 1-\epsilon + \frac{\epsilon}{m} & \text{if } a = \arg \max_{a'} Q^\pi(s, a') \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$ 적용

highest Q-Value를 갖는 action $\arg \max_a Q^\pi(s, a')$ 에 대해 $1-\epsilon$ 의 확률을 가지고 있음

그러나 highest Q-Value를 갖는 action $\max_a q_\pi(s, a)$ 대해서 $(1-\epsilon)$ 확률을 갖도록 적용. $(1-\epsilon) \max_a q_\pi(s, a)$

그 다음 모든 action에 대해 $\frac{\epsilon}{m}$ 을 전부 더해주었음. 그래서 모든 action \sum_a 에 대해서 $\frac{\epsilon}{m}$ 을 공유

$$- \frac{\epsilon}{m} \sum_a q_\pi(s, a) + (1-\epsilon) \max_a q_\pi(s, a) \geq \frac{\epsilon}{m} \sum_a q_\pi(s, a) + (1-\epsilon) \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s, a)$$

$(\because \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} = 1)$ \therefore 이전 policy $\pi(a|s)$ 가 있음. π 는 ϵ -greedy policy π' 를 사용. $\pi(a|s) = \begin{cases} 1-\epsilon + \frac{\epsilon}{m} & \\ 0 & \end{cases}$

방법으로 구한 policy. 모든 action \sum_a 에 대해서 $-\frac{\epsilon}{m}$ 을 빼줌. $\pi'(a|s)$ 적용시 $\pi'(a|s) = \begin{cases} 1-\epsilon + 0 & \\ 0 & \end{cases}$

highest Q-Value를 갖는 action에 대해서 $a = \arg \max_{a'} Q^\pi(s, a')$ 는 $1-\epsilon$ 이고 나머지는 0.

$\sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} \leftarrow$ highest Q-Value만 $1-\epsilon$, 나머지 0 = $1-\epsilon + 0 + 0 + 0 + \dots + 0$

결국 $\frac{1-\epsilon}{1-\epsilon} = 1$. 확률과 같은 모음을 가지고 있음

$\sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s, a)$ 보면 $q_\pi(s, a) + \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon}$ 확률 같은 모음을 sum을 해주었는데 $\max_a q_\pi(s, a)$ 는 highest Q-value를 바로 선택해준 것. 그래서 항상 $\max_{a'} q_\pi(s, a) \geq \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s, a)$

$$- \frac{\epsilon}{m} \sum_a q_\pi(s, a) + (1-\epsilon) \sum_a \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s, a) = \sum_a \pi(a|s) q_\pi(s, a)$$

$$- \sum_a \pi(a|s) q_\pi(s, a) = V_\pi(s) \quad V_\pi(s) \text{의 정의}$$

결론 $q_\pi(s, \pi'(s)) > V_\pi(s)$ 보다 항상 크거나 같다. $q_\pi(s, \pi'(s)) \geq V_\pi(s)$

Policy 보장 • For any ϵ -greedy π , ϵ -greedy policy π' w.r.t. q_π is always improved

Improvement

Theorem $V_{\pi'}(s) \geq V_\pi(s)$

설명 ϵ -greedy policy를 사용해도 policy를 improve되는 보증.

DP와 RL(MC)의 큰 차이점은 greedy policy가 아닌 ϵ -greedy policy $\pi'(a|s) = \begin{cases} 1-\epsilon + \frac{\epsilon}{m} & \\ \frac{\epsilon}{m} & \end{cases}$ 사용

터 이상 deterministic이 아닌 stochastic policy를 사용.

Q _t	↑	↓	←	→
0	0.1	-	-	-
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-

Greedy in Limit with Infinite Exploration (GLIE)

시간이 흐를수록 π 값을 0으로 만들기 위해 행동

GLIE 설명

Learning policy is called GLIE. if it satisfies: 행동 policy가 아래 두 조건 만족

Infinite 조건

- all state-action pairs (s, a) are explored infinitely many times.

Exploration

$$\lim_{K \rightarrow \infty} n_K(s, a) = \infty$$

설명 1 - 모든 state-action pair에 대해서 가진 샘플들이 무한히 많이 반복적으로 방문해야 함

$n_K(s, a)$ 는 increment count. state-action pair를 방문하는 횟수를 나타냄. K 는 K 번째 샘플

$K \rightarrow \infty$ 란 의미는 무한히 많은 에피소드를 선택할 때, state-action pair를 방문하는 횟수가 모든 pair에 대하여 무한히 많아져야 한다. = Infinite Exploration

Greedy 조건 2

the learning policy converges to greedy policy

in limit

$$\lim_{K \rightarrow \infty} \pi_K(a|s) = 1 \text{ where } a = \arg \max_a Q_K(s, a)$$

(1-8+10)

설명 2: ϵ -greedy policy를 적용할 경우 성능이 좋지 않아 해결책: ϵ 값을 점점 줄여 greedy로 $\pi_K(a|s)$ ϵ -greedy policy로 시작해 $K \rightarrow \infty$ 궁극적으로 $a = \arg \max_a Q_K(s, a)$ highest Q-value를 갖는 best action에 대해서는 $1 - \epsilon + \frac{\epsilon}{m}$ 을 적용하기로 했음. 궁극적으로는 π 를 점점 작게 해서 1로 가게 함 limit으로 갈 때는 policy가 Greedy로 바뀌어야 한다. = Greedy in Limit

실제로는

조건 1은 data sample을 최대한 늘리는 것 외에 방법이 없음.

조건 2는 ϵ 값을 조절하면 가능. 그

GLIE MC control: ϵ -greedy is GLIE if $\epsilon_K \rightarrow 0$ as the following

- in k -th sample (episode) using π , for each state-action pair (S_t, A_t) ,

$$n(S_t, A_t) \leftarrow n(S_t, A_t+1), Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{n(S_t, A_t)} [G_t - Q(S_t, A_t)]$$

설명: ϵ_K 값을 k -th sample들이 점점 늘어날 때마다 ϵ_K 값을 0으로 만들게 만들면 $\epsilon_K \rightarrow 0$, ϵ -greedy는 GLIE 양쪽 각 state-action pair (S_t, A_t) 에 대해서, sample을 계속 늘리게 되면 $n(S_t, A_t)$ 가 점점 커지며 그것을 사용해 update: $Q(S_t, A_t) + \frac{1}{n(S_t, A_t)} [G_t - Q(S_t, A_t)] \leftarrow$ Incremental MC update

실제로 Constant \rightarrow MC를 적용할 수 있음

- improve policy based on new action-value function.

$$\epsilon = \frac{1}{K} \text{ and } \pi \leftarrow \epsilon\text{-greedy}(Q)$$

설명: 만약 ϵ 를 $\frac{1}{K}$ 로 하면 학습이 진행될 때마다 $\frac{1}{K}$ 은 0으로 수렴. ϵ -greedy에서 ϵ 가 0으로 수렴하기에 greedy policy로 쓰임

GLIE MC Control converges to the optimal: $Q(s, a) \rightarrow q_*(s, a)$

설명: GLIE MC Control을 쓰면 optimal 값으로 Converges하게 됨.

Q -value estimate $Q(s, a)$ 가 true optimal Q -value $q_*(s, a)$ 로 도달함.

Monte Carlo method 수도 코드

- Initialize $Q(s, a)$, all $s \in S, a \in A(s)$, arbitrarily and $Q(\text{terminal}, \cdot) = 0$

Returns(S, A) \leftarrow empty list

$\pi \leftarrow$ arbitrarily ϵ -soft policy (non-empty probabilities)

12月

모든 State-action pair all $S, A \in \mathcal{A}(S)$ 에 대해서 Q-Value $Q(S, A)$ 들을
arbitrarily(random)하게 initialisation!, terminal state 값 $Q(\text{terminal})$ 초기화
모든 State-action pair에 대해 return 값을 계산해야 함 Returns(S, A), return 값을 empty list 초기화
policy π로 처음에는 arbitrarily(random)하게 고름. ϵ -soft policy: arbitrarily하게 하되.
모든 policy 값이 0은 안되게 하겠다. 모든 state에 대해서 action 확률이 0이 되는 것은 하지 않겠다.
정도로 아주 작은 숫자 보다는 더 큰 확률로 action을 선택하겠다는 의미. (non-empty probabilities)

- Repeat forever (for each episode) :-

설명: episode 하나가 주어지면, Policy Evaluation와 Policy Improvement를 적용함.

- (a) Generate an episode using $\pi : S_0, A_0, R_1, S_1, \dots, S_{T-1}, A_{T-1}, R_T$

$$G \leftarrow 0$$

설명: episode trajectory 가시도: $S_0, A_0, R_1, S_1, \dots, S_{T-1}, A_{T-1}, R_T$

현재까지 주어진 policy π를 가지고 실제 환경에서 action을 취해 얻어진 data samples

return G^2 0으로 초기화

- MC Policy (b) Repeat (for each step of episode), $t=T-1, T-2, \dots, 0$ \circ

Evaluation

$$G \leftarrow \gamma G + R_{t+1}$$

- (prediction) Unless (S_t, A_t) appears in $(S_0, A_0), \dots, (S_{t-1}, A_{t-1})$: first visit MC

Append G_t to Returns(S_t, A_t)

성명

episode 안에 각 step에 대해 (b)를 반복. 즉 $(S_0, A_0, R_0), (S_1, A_1, R_1)$ pair 를 각각 반복.

이미 전체데이터(episodic trajectory)를 가지고 있을 때부터 RT까지 값을 가지고 있을

time step을 뒤에서부터 앞으로 돌아 $t=T-1, T-2, \dots, 0$: return 값을 계산하기 편리 하려고.

보이우: 처음 $G=0$, $G \leftarrow 0$ + 에서 $T-1$ \uparrow $G \leftarrow \gamma G + R_{t+1} = \gamma \times 0 + R_{t-1} + 1 = R_t$

t 가 $T-2$ 일 때 $G \leftarrow \gamma R_t + R_{t-1}$ / t 가 $T-3$ 일 때 $G \leftarrow \gamma(\gamma R_t + R_{t-1}) + R_{t-2}$...

기본적으로 $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1} R_T$ 가 됨. t 는 timestep에서 계산방법

State-action pair (S_t, A_t) 가 이전에 $(S_0, A_0), \dots, (S_{t-1}, A_{t-1})$ 나오지 않았다면, (처음)

Returns (S_t, A_t) 변수에 봄금 계산한 return G_t 를 추가함.

위방법이 First Visit Monte Carlo 방법임.

Returns (S_t, A_t) 값을 이용하여 mean return value (average Returns (S_t, A_t))을 계산

ϵ -greedy (c) For each S_t in the episode:

Policy $A^* \leftarrow \arg \max Q(S_t, a)$

Improvement For all $a \in A(S_t)$:

(control) $\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(S)|} & \text{if } a = A^* \\ \frac{\epsilon}{|A(S)|} & \text{otherwise} \end{cases}$
action π 수

설명: episode 안에 있는 State S_t 에 대해서

Samples state S_t 에서 highest Q-Value를 A^* 라고 함

highest Q-Value를 갖는 $(a \in A)$ 에 대해 $1 - \epsilon + \frac{\epsilon}{|A(S)|}$ 의 확률을 적용하고

나머지에 대해서는 $\frac{\epsilon}{|A(S)|}$ 의 확률을 적용함

위 방식으로 $\pi(a|S_t)$ 를 적용함

결론 (a), (b), (c)를 반복적으로 적용하다보면 PE와 PI가 Stabilize' 할 때가 나옴. 학습 종료