Monte Carlo Methods

Agenda

- Last : Dynamic Programming – Known MDP
- This : Monte Carlo Method (a.k.a. MC)
  - Prediction
  - Estimation of Action Values
  - Control
  - Control w/o Exploring Starts
  - Off-policy (prediction, MC Control)
  - Incremental Implementation

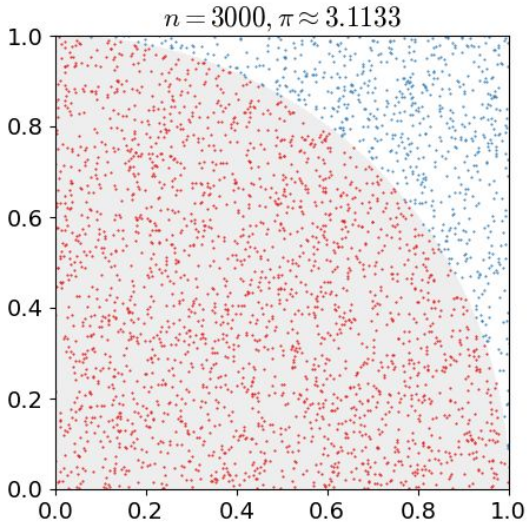
Monte Carlo Method?

- 동명의 카지노에서 따온 이름을 가진, 무작위 추출된 난수를 이용하여 함수의 값을 계산하는 통계학의 방법. 최적화, 수치적분, 확률분포로부터의 추출 등에 쓰인다.

https://namu.wiki/w/%EB%AA%AC%ED%85%8C%20%EC%B9%B4%EB%A5%BC%EB%A1%9C%20%EB%B0%A9%EB%B2%95

Monte Carlo Method Example

- 사분원을 이용한 원주율 계산 방법
  - 정사각형 내 사분원
  - 점을 균일하게 분포
  - 개수 세어 원주율 구함



Monte Carlo Method

- Require only experience
  - Actual experience : no period knowledge
  - model-free: no knowledge of MDP transitions / rewards
- Define MC method that..
  - Only for episodic task : episode by episode
  - Experience is divided into episodes
  - All episodes eventually terminate
    - Episode 완료하여 값 예측하고 정책을 바꾼다.

Monte Carlo Prediction

- State value function 을 학습
  - 동일한 state 에서 발생한 return 을 확인, 평균 함
- First-visit MC, every-visit MC
  - First-visit MC : s 로 처음 방문한 결과를 평균
  - Every-visit MC : s 로 방문하는 모든 결과를 평균
    - Every-visit MC : ch9, ch12 에서 볼 예정.

Monte Carlo Prediction

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

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

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

MC Prediction - Blackjack

- Rule
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a “useable” ace? (yes-no)
- State : Hit(continue) / Stick(stop) / Bust
- Reward : +1 / -1 / 0 (win/lose/draw)
- Our policy : stick if sum is 20 or 21

MC Prediction - Blackjack

- 규칙

4.1. 기본 규칙 [편집]

룰은 먼저 베팅 금액을 정한다. 배당율은 건 금액만큼을 받는 게 기본. 100원을 걸어 이기면 200원을 받고[19] 지면 건 금액인 100원을 잃는다.

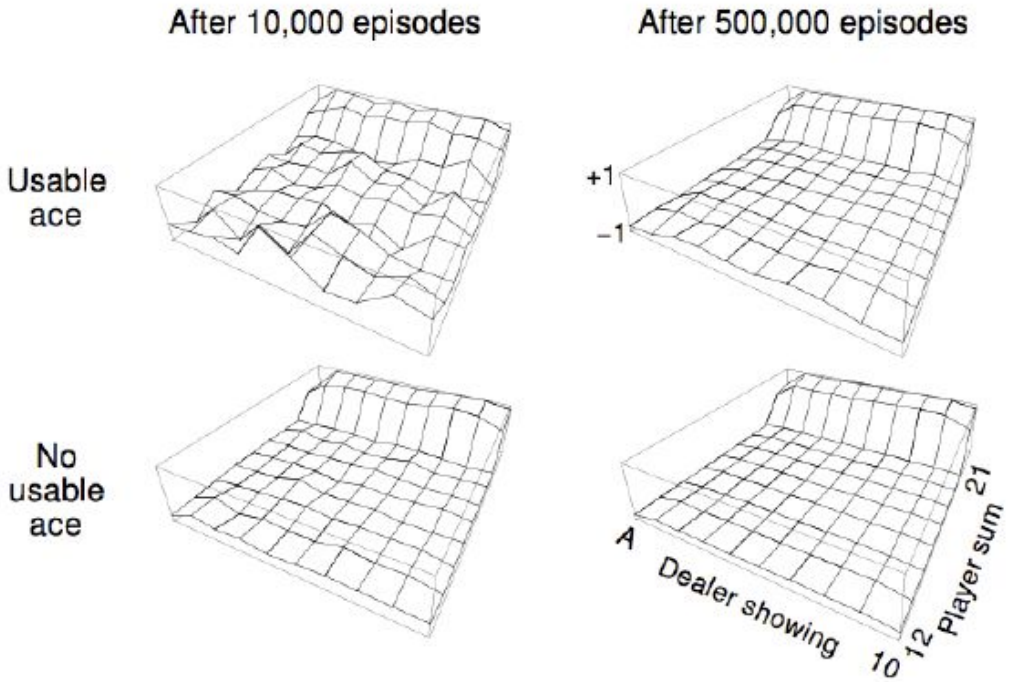
카드의 숫자 계산은 카드에 써 있는 숫자 그대로. 이 숫자를 더해서 21을 만들면 되는 간단한 게임이다. K, Q, J는 10에 해당하며, A는 1 혹은 11 어느 쪽으로도 계산할 수 있다. 하지만 일부 카지노에서는 1로만 적용한다든지 11로만 적용한다든지 카지노에 따라 다양한 룰을 적용하고 있으니 반드시 하기 전에 확인할 것.[20]

카드 두 장을 기본적으로 지급받게 되며, 카드 숫자를 합쳐 가능한 21에 가깝게 만들면 이기는 게임. 처음 받은 2장 합쳐 21이 나오는 경우 블랙잭이 되며[21][22] 21이 되지 않았을 경우 원한다면 얼마든지 카드를 계속 뽑을 수 있다. 하지만 카드 숫자의 합이 21을 초과하게 되는 순간 '버스트'라고 하며 딜러의 결과에 관계없이 플레이어가 패배한다.[23]

더블다운 서렌더 같은 추가적인 룰들이 있으니 해당 카지노의 규칙을 알아보고 가는 것이 좋다.[24]

https://namu.wiki/w/%EB%B8%94%EB%9E%99%EC%9E%AD(%EC%B9%B4%EB%93%9C%EA%B2%8C%EC%9E%84)

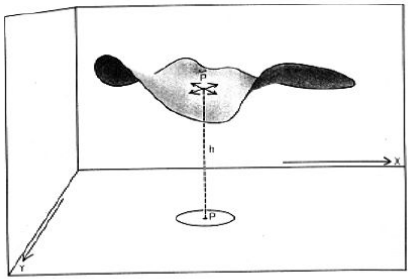
MC Prediction - Blackjack



MC Prediction - Blackjack

- DP vs MC
  - DP diagram shows all possible transitions
  - DP 는 one step transaction 을 transition diagram 에 나타내지만, MC 는 한 episode 내 전부를 확인할 수 있음.
  - DP 가 모든 경우 (history) 를 보고 추적해야 하지만, MC 는 서로 독립적으로 예측(estimation) 함.

MC Prediction - Soap Bubble



A bubble on a wire loop.

- A bubble on a wire loop
  - 목적 : 표면 구하기 (shape of surface)
  - 속성 : 표면의 모든 방향의 힘의 합은 0 (아님 터짐)
    - 한 지점의 표면 높이는 주변 평균
- 일반적 해법
  - 격좌를 그림
  - 특정점 주변 좌표 높이를 구하여 평균하여 높이구함
- 중간부터 값 구하더라도 가능 (DP 와 차이점)

MC Estimation of Action Value

- Model 이 없으면 action value 예측 더 유용함
  - Model 이 있으면 state value 만으로 예측 가능
  - Model 이 없으면 state value 만으로 예측 불가
    - Action value를 이용하여 policy를 추정함.
- State-action pair (s, a) 를 추정
  - Every-visit MC 에서는 방문한 모든곳 평균
  - First-visit MC 에서는 Episode 의 첫 pair 평균

MC Estimation of Action Value

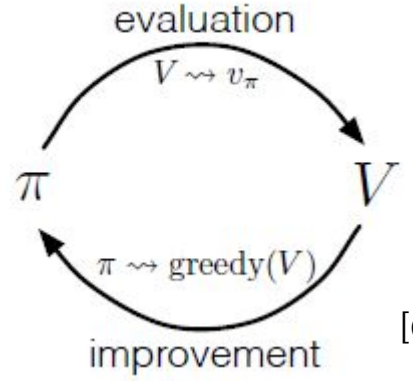
- 문제점 (“maintaining exploration”)
  - “결정론적 정책(Deterministic Policy)”은 단일 state, 단일 Action 연결되는데, 학습적 개선 어려워 심각한 문제가 됨.
    - Policy 가 결정론적 정책이면 정책을 따르는 것은 각 state 마다 하나의 동작(action) 임
    - 평균 할 반환값 없으면 예측이 개선되지 않음
    - Action Value 를 배우는 것은 선택 돕게 하려는 것인데 이 목적이 달성되지 않음.

MC Estimation of Action Value

- Maintaining exploration 해결법
  - Action value 를 위해 동작하도록 policy를 평가할 때 state 를 무한히 방문함 가정 "State-action pairs will be visited an infinite number of times"
  - 다음 몇가지를 가정 ("Exploring starts")
    - Episode 는 state-action pair 에서 시작
    - 모든 pair 는 시작에서 선택될 때 확률이 0이 아님
    - State-action pair 가 무한개 episode 에서 무한번 방문 보장

Monte Carlo Control

- MC Estimation 이 최적해 근사하는 방법
- GPI (Generalized Policy Iteration) 유사 (Ch. 4.6)
  - Policy Evaluation, Policy Improvement 반복



[GPI in Ch. 4.6.]

$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*,$$

[Policy Evaluation & Policy Improvement]

Monte Carlo Control

- Policy 는 greedy 하게 사용
- Policy Improvement

$$\pi(s) \doteq \arg\max_a q(s,a).$$

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$

$$q_\pi(s, \pi'(s)) \geq v_\pi(s). \quad (4.7)$$

$$v_{\pi'}(s) \geq v_\pi(s). \quad (4.8)$$

Monte Carlo Control

- 실사용 위해서는 Maintaining exploration (p.14) 위한 가정 제거
  - 무한 방문, exploring starts
- 무한방문 제거 방법
  - Approximating $q_{\pi k}$ in Each Policy Evaluation
    - Some level of approximation : 수준의 상한 설정
  - Avoiding the infinite number
    - Give up trying : 무한대 반복하지 않고 멈추기
  - 적절한 방법? : 충분히 수렴 못할 경우 / 줄인 횟수가 적절한지 검토 필요

[참고]
위에서 얘기한 policy evaluation과 policy improvement는 Monte Carlo policy iteration으로 본다면 매 step마다가 아니라 episode가 끝날 때마다 하는 것으로 바꿀 수 있다. 에피소드가 끝나면 return의 값으로 policy evaluation을 하고 이것이 끝나면 policy를 개선하고 끝나면 다시 에피소드 시작하는 것이다. 아래 의사 코드는 이를 반영한 Monte Carlo ES(Exploring Starts)이다.

Monte Carlo Control

First-visit MC prediction, for estimating $V \approx v_\pi$

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Expected -> optimal value

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
Append G to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

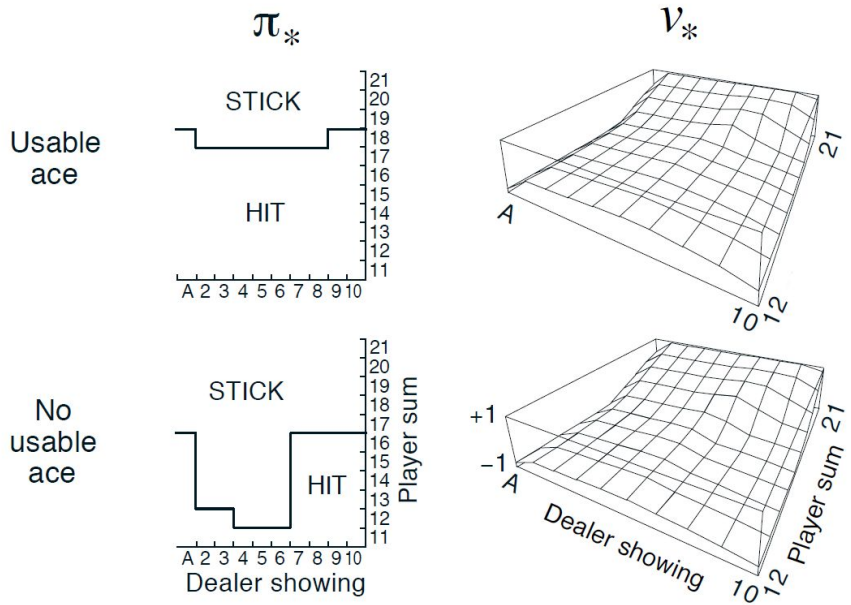
변경점
A(s) 추가

모든 경우 대신 임의 경우?

$\mathcal{A}(s)$	set of all actions available in state s
Q, Q_t	array estimates of action-value function q_π or q_*

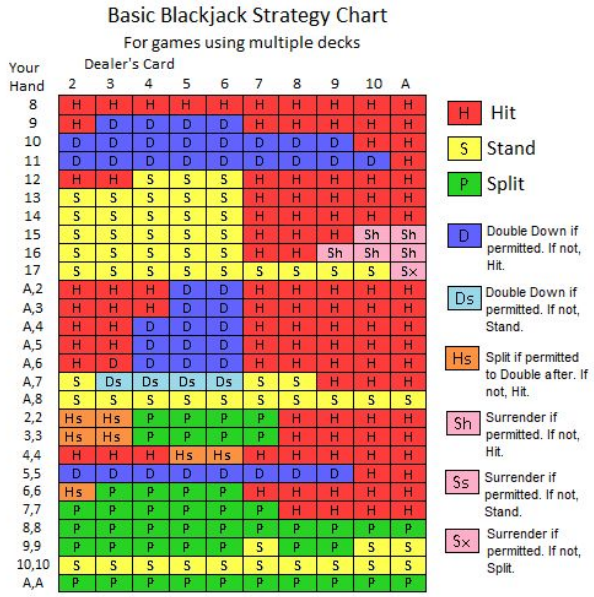
Monte Carlo Control

- Solving Blackjack (with MC ES)



Monte Carlo Control

- (참고) Blackjack Strategy Chart



2. 설명 [편집]

딜러에게 카드를 한장씩 받아 21에 가까운 수를 만드는 사람이 이기며 21을 초과하면 지는 게임. 기본 룰은 간단하지만, 딜러와 하는 카드 게임 중에서는 실력에 따른 승률 편차가 상당히 크다.[2] 밑에서 서술할 룰들을 플레이어에게 가장 최선의 전략만 선택하여[3] 플레이 했을 때 이론상의 승률은 세부 룰에 따라 다르지만 보통 44%~49.5%다. 44%에 가까운 경우는 플레이어한테 유리한 세부 룰을 거의 적용 안 한 경우[4]이며 49.5%는 카지노가 불리해지지 않는 선에서 최대한 세부 룰을 적용해 주었을 경우[5]에 해당한다.

실제로 여러 세부 규칙들은 상황에 따라 플레이어가 사용 여부를 결정할 수 있으므로 플레이어에게 유리한 규칙이다. 딜러는 8, 8이 나오더라도 스플릿 할 권한 없이 16으로 게임을 진행해야 되고 처음 두 카드의 합이 11이 나오더라도 더블 다운을 할 권한이 없다. 하지만 이러한 룰을 카지노가 허용해도 환수율이 100% 미만으로 나오는 이유는 플레이어가 버스트 되면 딜러가 무조건 이긴다는 점이 크기 때문이다.

세부 규칙에 따라 다르지만 1덱 핸드셔플, 서렌더 허용, AA 스플릿 후 또 A가 나왔을 시 다시 스플릿 허용, 모든 경우에 더블다운 허용 등 플레이어에게 유리한 규칙을 모두 적용하면 플레이어 환수율이 100%를 넘겨버리기 때문에 이론상 게임이 누적될 수록 카지노는 굉장한 손해를 보게 된다. 만약 1000원 게임 한 판 할 때마다 카지노의 기대수익이 -20원이면[6] 플레이어 개인별로는 이득 보는 플레이어도 있고 손해 보는 플레이어도 있겠지만 카지노 입장에서는 장기적으로는 플레이어들에게 손해 보는 구도로 흘러간다[7] 그래서 저 모든 세부 룰을 적용하는 자비로운 카지노는 세상에 존재하지 않는다. 당장 2덱 핸드셔플을 하는 유럽 카지노에서는 서렌더를 허용하지 않고, 서렌더를 허용하는 홍콩의 카지노에서는 덱을 여러 개(6덱이나 8덱) 사용한다. 카지노마다 다르므로 룰을 미리 확인해보는 것도 좋은 방법이다.

검색 : Blackjack strategy chart / 나무위키 블랙잭

MC Control without ES

- ES 를 제거하려면 계속 탐험하도록!
  - On-policy : 결정한 행동으로 정책 평가, 개선
  - Off-policy : 만들어진 데이터로 정책 평가, 개선
- On-policy method : ε-greedy policies (epsilon greedy)
  - Most of time, maximal estimated action value
  - ε 확률로 random 선택
    - Greedy 선택 확률 : $1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|}$ Greedy 확률 + random 에서 greedy 선택가능 확률
    - 임의의 Non Greedy 선택확률 : $\frac{\varepsilon}{|\mathcal{A}(s)|}$ 확률 / 선택가능 수

$\mathcal{A}(s)$ set of all actions available in state s

MC Control without ES

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

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

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

MC Control without ES

- Policy improvement with epsilon greedy

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s,a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\varepsilon) \max_a q_\pi(s,a) \qquad (5.2) \\
&\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1-\varepsilon} q_\pi(s,a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) - \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + \sum_a \pi(a|s) q_\pi(s,a) \\
&= v_\pi(s).
\end{aligned}
$$

Random select

Greedy select

MC Control without ES

- Bellman optimality equation (3.19) 에 적용

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_*(s')\big]. \quad (3.19)$$

Optimal Value Function
On epsilon-soft policy

$$\begin{aligned}\tilde{v}_*(s) &= \max_a \sum_{s',r}\Big[(1-\varepsilon)p(s',r|s,a) + \sum_{a'}\frac{\varepsilon}{|\mathcal{A}(s)|}p(s',r|s,a')\Big]\Big[r+\gamma\tilde{v}_*(s')\Big] \\ &= (1-\varepsilon)\max_a \sum_{s',r} p(s',r|s,a)\Big[r+\gamma\tilde{v}_*(s')\Big] \\ &\quad + \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a\sum_{s',r} p(s',r|s,a)\Big[r+\gamma\tilde{v}_*(s')\Big].\end{aligned}$$

Policy π 가 Optimal Policy 일 경우

$v_\pi(s) = \tilde{v}_*(s)$

Epsilon-soft policy π 가
더 이상 개선이 없을 경우 5.2 로 부터..

$$\begin{aligned}v_\pi(s) &= (1-\varepsilon)\max_a q_\pi(s,a) + \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a) \\ &= (1-\varepsilon)\max_a \sum_{s',r} p(s',r|s,a)\Big[r+\gamma v_\pi(s')\Big] \\ &\quad + \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a\sum_{s',r} p(s',r|s,a)\Big[r+\gamma v_\pi(s')\Big].\end{aligned}$$

$$q_\pi(s,\pi'(s)) = \frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a) + (1-\varepsilon)\max_a q_\pi(s,a) \quad (5.2)$$

$v_\pi(s)$	value of state s under policy π (expected return)
$v_*(s)$	value of state s under the optimal policy
$q_\pi(s,a)$	value of taking action a in state s under policy π

-> Exploring starts 제거

Off-policy Prediction via Importance Sampling

- On-policy 를 통하여 최적 유사한 policy 탐색 (Near Optimal Policy)
- 두개의 정책을 사용하는 방법
  - Optimal Policy 를 학습하여 사용 : target policy : π (Learn about and becomes the optimal policy)
  - 행동을 만듬 : behavior policy : b (Generate behavior)
- Off-policy learning : Target policy 가 아닌 data 에 의하여 배우는 방법
  - Additional concept, notation 필요하고 학습 느리지만 더 다양하게 사용 가능

Off-policy Prediction via Importance Sampling

- 예측에 사용하므로 다음 가정
  - b, π : both fixed and given
  - At all episode, another policy with b ≠ π
- Assumption of coverage
  - π 에서 선택된 모든 action 은 b에서도 취해져야 함
  - b 는 π 에 없더라도 확률적일 수 있음 (like ε-greedy)
- Prediction Problem 이기에 π 는 unchanging, given 으로 가정

Off-policy Prediction via Importance Sampling

- Importance-sampling ratio
  - 대부분 off-policy 에서 기대값을 예상하기 위해 중요도 샘플링을 수행
  - State-action 의 trajactory $A_t, S_{t+1}, A_{t+1}, \ldots, S_T$ 의 발생확률

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}$$

P : state-transition probability function (3.4)
상태 전이 확률

$$p(s'|s,a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a). \qquad (3.4)$$

Off-policy Prediction via Importance Sampling

- Importance-sampling ratio

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k,A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k,A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}.$$

- Trajectory probability 가 MDP 의존적이지만, importance sampling ratio 는 두개의 정책으로 정리 됨

Off-policy Prediction via Importance Sampling

- Estimate the expected return
  - Behavior policy 로 Gt 만 가짐
  - Importance sampling 적용하여 ratio 를 넣어 적절한 기대값 예측에 사용

$$\mathbb{E}[G_t | S_t = s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s).$$

Off-policy Prediction via Importance Sampling

- Ordinary importance sampling

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}.$$

Pt : t 에 의한 처음 종료
T(t) : t 시점의 처음 종료
Gt : t 시점의 T(t) 에 의한 반환값(return)

방문한 적이 있는 모든 s에 대한 timestamp 합

- Weighted importance sampling

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}},$$

혹은, 분모 0 일 경우 0

Off-policy Prediction via Importance Sampling

- Ordinary vs Weighted importance sampling
  - Ordinary importance sampling : 비편향, 분산 커짐
  - Weighted importance sampling : 편향, 분산 적음
  - 실제로는 분산 적어서 Weighted 많이 사용함.

Incremental Implementation

- Chapter 2 에서와 같이 Incremental Implementation 사용 가능
  - Averaged reward 대신 averaged return 사용
  - On-policy 는 동일하게 사용
  - Off-policy 의 경우 scaled return 사용 위하여 weighted importance sampling 사용

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \qquad n \geq 2,$$

W : Weight
G : Sequence of returns (G1, G2, ...)

Off-policy Prediction via Importance Sampling

- Commulative sum Cn 으로 정리

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}\Big[G_n - V_n\Big], \qquad n \geq 1,$$

$$C_{n+1} \doteq C_n + W_{n+1},$$

Off-policy Prediction via Importance Sampling

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Off-policy MC Control

- On-policy : control 위해 policy value 추정
- Off-policy : target policy, behavior policy
  - Target policy 가 deterministic 하더라도 behavior 가 ε-soft 하여 최적으로 수렴 가능

Off-policy MC Control

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
$b \leftarrow$ any soft policy
Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Non Greedy 가 일반적(common)이게 되면 학습 속도가 느려짐
γ 가 1 보다 작은 경우 다음 section 에서 개선 가능

Discounting-aware Importance Sampling

- Discount 를 고려하여 (Discounting-aware) Importance Sampling을 수행
- Discount 대상 :
  - 부분적으로 종료 된 정도에 따라서 (termination, equivalently, partial termination
- Flat partial return : discount 없는 구간

$$\bar{G}_{t:h} \doteq R_{t+1} + R_{t+2} + \cdots + R_h, \qquad 0 \leq t < h \leq T,$$

Discounting-aware Importance Sampling

Full Result

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

전개하면 ..

$$\begin{aligned}
&= (1-\gamma) R_{t+1} \\
&\quad + (1-\gamma)\gamma \left(R_{t+1} + R_{t+2}\right) \\
&\quad + (1-\gamma)\gamma^2 \left(R_{t+1} + R_{t+2} + R_{t+3}\right) \\
&\quad \vdots \\
&\quad + (1-\gamma)\gamma^{T-t-2} \left(R_{t+1} + R_{t+2} + \cdots + R_{T-1}\right) \\
&\quad + \gamma^{T-t-1} \left(R_{t+1} + R_{t+2} + \cdots + R_T\right)
\end{aligned}$$

이를 Flat partial return 으로 정리

$$= (1-\gamma) \sum_{h=t+1}^{T-1} \gamma^{h-t-1} \bar{G}_{t:h} \quad + \quad \gamma^{T-t-1} \bar{G}_{t:T}.$$

Flat partial return

Discounting-aware Importance Sampling

- Ordinary importance-sampling estimator 적용

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \left((1-\gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \bar{G}_{t:h} \;+\; \gamma^{T(t)-t-1} \rho_{t:T(t)-1} \bar{G}_{t:T(t)} \right)}{|\mathcal{T}(s)|}, \quad (5.9)$$

- weighted importance-sampling estimator 적용

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \left((1-\gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \bar{G}_{t:h} \;+\; \gamma^{T(t)-t-1} \rho_{t:T(t)-1} \bar{G}_{t:T(t)} \right)}{\sum_{t\in\mathcal{T}(s)} \left((1-\gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \;+\; \gamma^{T(t)-t-1} \rho_{t:T(t)-1} \right)}. \quad (5.10)$$

Per-decision Importance Sampling

- Ordinary / weighted importance-sampling 의 분자 부분이 동일함

$$\begin{aligned}\rho_{t:T-1}G_t &= \rho_{t:T-1}\left(R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1}R_T\right) \\ &= \rho_{t:T-1}R_{t+1} + \gamma\rho_{t:T-1}R_{t+2} + \cdots + \gamma^{T-t-1}\rho_{t:T-1}R_T. \qquad (5.11)\end{aligned}$$

- 처음것과 마지막 것만 의미 있음(reward 연관)

$$\mathbb{E}\left[\frac{\pi(A_k|S_k)}{b(A_k|S_k)}\right] \doteq \sum_a b(a|S_k)\frac{\pi(a|S_k)}{b(a|S_k)} = \sum_a \pi(a|S_k) = 1.$$

$$\mathbb{E}[\rho_{t:T-1}R_{t+1}] = \mathbb{E}[\rho_{t:t}R_{t+1}]\,.$$

$$\mathbb{E}[\rho_{t:T-1}R_{t+k}] = \mathbb{E}[\rho_{t:t+k-1}R_{t+k}]\,.$$

Per-decision Importance Sampling

- Per-decision importance sampling

$$\mathbb{E}[\rho_{t:T-1} G_t] = \mathbb{E}\left[\tilde{G}_t\right],$$

where

$$\tilde{G}_t = \rho_{t:t} R_{t+1} + \gamma \rho_{t:t+1} R_{t+2} + \gamma^2 \rho_{t:t+2} R_{t+3} + \cdots + \gamma^{T-t-1} \rho_{t:T-1} R_T.$$

Per-decision Importance Sampling

- Ordinary importance sampling 에 Gt 사용
  - Ordinary importance sampling 을 수행하면서 분산은 줄임

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \tilde{G}_t}{|\mathcal{T}(s)|},$$

Summary

- Monte Carlo 특징
  - Sample episode 의 경험을 통해 Value function, optimal policy 개선
  - 모델 없이 환경과 상호작용으로 최적 행동 학습
  - 시뮬레이션이나 샘플 모델 사용 가능 (model free)
  - 충분한 탐색을 위한 ES : Exploring starts
  - 자신의 정책 사용 (on-policy) 외 behavior policy 로 target policy 찾는 off-policy

Summary

- Monte Carlo 특징
  - Off-policy 에서 다른 정책 사용하기 위해 sampling
    - Ordinary importance sampling (unbiased, large variance)
    - Weighted importance sampling (biased, finite variance, preferred)

Thanks!

Any questions ?