* MDP (Markov Decision Process)

: 시간 t 에서의 상태는 t-1에서의 상태에만 영향을 받음 (first-order Markov assumption)

- S: state, $P(s'|s)$: $s \to s'$로 이동할 확률 (transition probability), R: reward ($r(s) = E[R_{t+1} | S_t = s]$), γ: discount factor (현재와 미래의 reward 사이의 중요도 조절)
- Return : G_t (시간 t 이후부터 얻을 수 있는 reward의 합).

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

- State value function : V(S), Bellman equation으로 나타냄.

$$V(s) = E[R_{t+1} + \gamma R_{t+2} + \cdots | S_t = s] = R_{t+1} + \gamma \sum_{s' \in S} P(s'|s) V(s')$$

⇒ MDP는 action이 요소가 추가됨

∴ policy (각 state 마다 action이 선택될 확률) 필요. : $\pi(a|s) = P(A_t = a | S_t = s)$

$$P_\pi(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s,a), \quad r_\pi(s) = \sum_{a \in A} \pi(a|s) r(s,a)$$

$$V_\pi(s) = E_\pi[G_t | S_t = s] = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s] = \sum_{a \in A} \pi(a|s)\left(r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_\pi(s')\right)$$

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s] = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \sum_{a' \in A} \pi(a'|s') q_\pi(a'|s')$$

- optimal state-value function $V_*(s) = \max_\pi V_\pi(s)$, optimal action-value function $q_*(s,a) = \max_\pi q_\pi(s,a)$

RL : learning policy when you do not have full info of rewards, dynamics

⇒ optimal policy를 formulate 할 수 없음

* Dynamic programming

: Bellman equation을 이용하여 주어진 완전한 MDP 환경에서 (↱ 환경에 대한 정보 필요) 모든 state의 value function을 이용해 간접적으로 optimal policy 계산.

Policy evaluation (prediction), Control을 반복하면서 optimal policy 구함.

Model based, Full width backup, Value based, off policy learning에 주로 쓰임

⟷ Policy optimization, Policy based, Policy gradient.

큰 문제 해결을 위해 작은 문제들로 나눔 (같은 문제에 대해서 항상 답이 같아야 함)

Bellman equation: Step별로 연산함. 각 step (state)에서의 value function은 저장되었다가 연산 시 다시 사용. optimal value function의 관계를 보여줌 (V(S), q(s,a) 모두 max 값 ↰)

+) backup : Bellman equation에서 V(s)와 q(s,a) 사이 연관성은 추가함. 현재의 value function은 미래의 value function으로 구함.

- boot strapping 사용. (Update estimates based on other estimates) [dynamic programming : full width backup / RL : sample backup]
- Policy evaluation ; backup without max (Prediction) → 현재 optimal 하지 않은 어떤 policy로 value function 구하는 것
- Policy improvement: from a greedy policy, at only locally

〈알고리즘〉

- Policy iteration : argmax 사용 (Control) bellman optimality eq 이용
- value iteration : back up에서 argmax가 아닌 max를 사용해 policy를 매번 update X. 주도 배움 (control) (↱ 현재의 value function을 토대로 더 나은 policy 구하는 것) bellman expectation eq 이용.
- 단점 : ① complexity ↑ ② state의 dimension이 늘어날 수록 state의 수가 exponential 하게 증가 ③ 환경에 대한 완전한 정보 (reward, state transition에 대한 확률) 필요.

ex. grid world.

* Monte Carlo Methods

: terminal state 까지 Step별로 얻은 reward를 모두 저장하고, terminal state에서 이를 바탕으로 state들의 value function 구함.

- DP에서는 bellman equation으로 value function을 구했다면, MC에서는 reward 들의 합의 평균 사용.

⇒ $V_\pi(s) \approx \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$, N(s): total episode 중 state s를 방문한 횟수, $G_i(s)$: episode i 에서 state s의 return 값

- 새로 return값을 받을 때마다 update : $V(s) \leftarrow V(s) + \frac{1}{n}(G(s) - V(s))$, $\frac{1}{n}$ = stepsize = α
- MC iteration의 문제점 : ① Value function의 model based ② Local optimum ③ Inefficiency of policy iteration ⇒ MC control : ① value function이 아닌 q-function 이용 ② ε greedy policy improvement 사용 ③ evaluation 한번만 진행.
- 단점: episode가 무한히 길면 update하기 어려움.

⇒ 환경에 대한 정보 없이 optimal Value function에 접근 가능. (Model-free)

- on policy / off policy
  - on-policy : 현재 상황에서 현재 best policy (behavior policy)로 행동했을 때 얻음. 자기 자신의 데이터만 사용, policy가 update되면 이전 policy 사용X. Q(s',a')
  - off-policy : 데이터를 자기 자신이 아닌 다른것으로부터 얻는 것 (ex. 과거의 자신 데이터), on policy 에 비해 신뢰성 떨어짐 Q(s',a)

* Temporal Difference method

: DP의 boot strapping + MC의 model free. 실제 reward와 다음 step에 대한 미래 추정 가치 사용 (다른 time point 사용) (이때 사용되는 reward + value function : TD target ↰)

- TD target : reward + value function. 하나의 step에서 계산하므로 noise↓. variance↓.
- TD error : 실제 V(s) − TD target.
- 각 단계가 끝나면 바로 update (MC에서는 전체 에피소드가 끝나야 보상을 나누어 단계별로 update)

⇒ $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$, $\alpha = \frac{1}{n}$

- Computational congerral

ex. SARSA (on-policy), Q-Learning (off-policy), expected SARSA (on policy, off policy), Double Q-learning (off-policy)

〈TD〉	vs. 〈MC〉
can learn online after every step	must wait until end of episode
can learn from incomplete sequence	can only learn from complete sequence
works in continuing (non-terminating) env	works for episodic (terminating) env.
low variance, some bias	high variance, zero bias
TD(0) converges to $V_\pi(s)$	good converge property
More sensitive to initial value	Not very sensitive to initial value

MC & TD converge: $V(s) \to V_\pi(s)$ as experience → ∞

experience → ∞ 가 아닌경우: [MC → solution with minimum mean squared errors. on-policy MC = ε soft policy로 수렴 / TD(0) → solution of max likelihood Markov model]

*	MDP	MC	TD
bootstrap	O	X	O
model	based	free	free
sampling	X	O	O
update	time-step	end of episode	time step

* 〈model free〉 vs 〈model based〉

〈model free〉	〈model based〉
환경에 대한 정보 X (경험을 통해 학습)	환경에 대한 정보 필요
sampling	sampling X (sample efficient)
Policy Gradient, Value learning	cost function
⇒ MC. TD	⇒ MDP

* N-step TD

: update 시 하나의 step이 아닌 n개의 step을 보고 update. (if n→∞ : MC)

- n 뿐만 아니라 step size도 영향을 줌. λ라는 weight 사용. (geometrically weighted average)

① forward-view TD(λ)

: n-step마다 모아진 return (G_t)에 weight 곱해 λ-return (G_t^λ) 만들어 update

⇒ $V(S_t) \leftarrow V(S_t) + \alpha(G_t^\lambda - V(S_t))$, $G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^n$

이때 weight는 $(1-\lambda)\lambda^{n-1}$ 형태를 이용해 전부 더하면 1이 됨.

→ episode가 끝나야 update 할 수 있다는 MC의 단점 재발생

② Backward-view TD(λ)

: eligibility trace를 기억해두고 이를 TD error에 곱해 update

↳ 과거에 방문했던 state 중 현재 얻게되는 reward에 영향을 주는 state를 판단해 현재 얻게되는 reward를 해당 state에 나눠줌. 기준 ① frequency heuristic ② Recency heuristic ∴ 멀리 떨어진 step일수록 return weight 낮아짐.

⇒ time-step마다 update 가능

* ε-greedy policy

- greedy : 학습하면서 얻은 value function의 값 중 가장 큰 값 따라감.

episode 별로 학습하는 경우 (TD, MC) 계속 같은 길로 가게되므로 exploration을 위해 ε-greedy policy 이용.

$$\pi(s) = \begin{cases} a^* = \arg\max_{a \in A} Q(s,a) & 1-\varepsilon \\ a \neq a^* & \varepsilon \end{cases}$$

⇒ ε의 확률로 가장 큰 Q(s,a)를 따라가지 않고 explore, 1−ε의 확률로 exploit

but, 계속 ε의 확률로 explore 하면 학습이 저하되므로 ε decay 이용 ($\varepsilon = \frac{\alpha}{i+1}$)

* explore vs exploit

- explore : Q가 높은 행동을 하는 것이 아닌 모든 action 중 random하게 골라 action.
- exploit : greedy 하게 Q가 높은 행동을 하는 것.

〈공식〉

* Propability

- $E[X] = \sum x P(x)$
- $\bar{X} = \frac{1}{n}\sum X_i$
- $E[\bar{X}] = E[\frac{1}{n}\sum X_i] = \frac{1}{n}\sum E[X]$

* MAB

- Q-estimates: $Q_t(a) = \dfrac{\sum_{i=1}^{t-1} R_i \mid A_i = a}{\sum_{i=1}^{t-1} 1 \mid A_i = a}$

* MDP

- return (G_t): $G_t = \sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$

$= R_{t+1} + \gamma G_{t+1}$

- Bellman expected Eq.

: $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a) = \sum_a \pi(a|s) \sum_{s'} [r(s,a,s') + \gamma V_\pi(s')] P(s'|s,a)$

$Q_\pi(s,a) = \sum_{s'} [r(s,a,s') + \gamma V_\pi(s')] P(s'|s,a) = \sum_{s'} [r(s,a,s') + \gamma \sum_{a'} Q_\pi(s',a') \pi(a'|s')] P(s'|s,a)$

- Bellman Optimality Eq.

$V^*(s) = \max_a Q^*(s,a) = \max_a \sum_{s'} [r(s,a,s') + \gamma V^*(s')] P(s'|s,a)$

$Q^*(s,a) = \sum_{s'} [r(s,a,s') + \gamma \max_{a'} Q^*(s',a')] P(s'|s,a)$

$\pi^*(a|s) = \begin{cases} 1 & a = \arg\max_{a'} Q^*(s,a') \\ 0 & \text{o.w} \end{cases}$

⇒ Policy evaluation : $V_{k+1}(s) \leftarrow \sum_a \pi_i(a|s) \sum_{s'} [r(s,a,s') + \gamma V_k(s')] P(s'|s,a)$

∴ $V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V_\pi(s')]$

$\pi(s) = \arg\max_a q(s,a)$

$q(s,a) = E_\pi\{G_t | S_t = s, A_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\}$

Value evaluation : $V_{k+1}(s) \leftarrow \max_a \sum_{s'} [r(s,a,s') + \gamma V_k(s')] P(s'|s,a)$

* pseudo code

MAB ① A simple bandit algorithm

```
Initialize, for a=1 to k :  Q(a) ← 0
                            N(a) ← 0   # of action a taken
Repeat forever: A ← { argmax_a Q(a)   with prob. 1-ε
                    { random action              ε
    R ← bandit(A)
    N(A) ← N(A)+1
    Q(A) ← Q(A) + 1/N(A) [R − Q(A)]
```

② Thompson sampling

: reward의 분포의 parameter를 확률 변수로 보고 random하게 추출한 값에 대해 가장 기댓값이 높은 arm 선택.

〈ε greedy〉

```
for t=1,2,... do
  for k=1,...K do
    θ̂_k ← α_k/(α_k+β_k)
  end for
  x_t ← argmax_k θ̂_k
  apply x_t and observe r_t
  (α_xt, β_xt) ← (α_xt + r_t, β_xt + 1 − r_t)
end for
```

〈Thomson Sampling〉

```
for t=1,2,... do
  for k=1,...K do
    Sample θ̂_k ~ beta(α_k, β_k)
  end for
  x_t ← argmax_k θ̂_k
  Apply x_t and observe r_t
  (α_xt, β_xt) ← (α_xt + r_t, β_xt + 1 − r_t)
end for
```

③ Policy Iteration

Initialize $V(s)$ arbitrarily;
Randomly initialize policy $\pi(s)$;
$\Delta \leftarrow 0$;
while $\Delta < \theta$ do — *Policy evaluation*
  for each $s \in S$ do
    $v \leftarrow V(s)$;
    $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))[r+\gamma V(s')]$;
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;
  end
end
Policy-stable $\leftarrow$ true; — *Policy Improvement*
for each $s \in S$ do
  old-action $\leftarrow \pi(s)$;
  $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r+\gamma V(s')]$;
  if old-action != $\pi(s)$ then
    Policy-stable $\leftarrow$ false;
  end
end
If policy-stable then
  return $V \approx v_*$ and $\pi \approx \pi_*$;
else
  go to policy evaluation;
end
end

④ Value Iteration

Initialize $V(s)$ arbitrarily, except $V(\text{terminal})$;
$V(\text{terminal}) \leftarrow 0$;
$\Delta \leftarrow 0$; (Loop until convergence)
while $\Delta < \theta$ do
  for each $s \in S$ do
    $v \leftarrow V(s)$;
    $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)[r+\gamma V(s')]$;
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;
  end
end
return $\pi, s_t, \pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r+\gamma V(s')]$;
end

⑤ On-policy MC control

Initialize, for all $s \in S$, $a \in A(s)$;
$Q(s,a) \leftarrow$ arbitrary
Returns$(s,a) \leftarrow$ empty list
$\pi(a|s) \leftarrow$ an arbitrary ε-soft policy
Repeat forever:
  Generate an episode using π
  For each pair s,a appearing in the episode:
    $G \leftarrow$ return following the first occurrence of s,a
    append G to Returns(s,a)
    $Q(s,a) \leftarrow$ average(Returns(s,a))
  For each s in episode:
    $A^* \leftarrow \arg\max_a Q(s,a)$
    For all $a \in A(s)$:
      $\pi(a|s) \leftarrow \begin{cases} 1-\varepsilon+\frac{\varepsilon}{|A(s)|} & \text{if } a = A^* \\ \frac{\varepsilon}{|A(s)|} & \text{if } a \neq A^* \end{cases}$

⑥ Off-policy MC prediction

Initialize, for all $s \in S$, $a \in A(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$
Repeat forever:
  $b \leftarrow$ any policy with coverage of π
  Generate an episode using b:
    $S_0, A_0, R_1, \cdots, S_{T-1}, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  For $t = T-1, T-2, \cdots$ down to 0:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
    If $W = 0$ then exit for loop

⑦ off-policy MC control

Initialize, for all $s \in S$, $a \in A(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$
Repeat forever:
  $b \leftarrow$ any soft policy
  Generate an episode using b: $S_0, A_0, R_1, \cdots, S_{T-1}, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  For $t = T-1, T-2, \cdots$ down to 0:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
    If $A_t \neq \pi(S_t)$ then exit for loop
    $W \leftarrow W \frac{1}{b(A_t|S_t)}$

⑧ TD(λ)

Initialize $V(s)$ arbitrarily (ex. $V(s) = 0$ for all $s \in S^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

⑨ SARSA (On-policy TD control)

Initialize $Q(s,a)$, $\forall s \in S$, $a \in A(s)$, arbitrarily and $Q(\text{terminal-state}) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (ex. ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
  until S is terminal

⑩ Q-learning (off-policy TD control)

Initialize $Q(s,a)$, $\forall s \in S$, $a \in A(s)$, arbitrarily and $Q(\text{terminal-state}) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (ex. ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A from S' using policy derived from Q
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
  until S is terminal

⑪ expected SARSA (on-policy / off-policy)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)]$$
$$\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma E(Q(S_{t+1}, A_{t+1}) | S_{t+1}) - Q(S_t, A_t)]$$

⑫ Double Q-learning (off-policy)

Initialize $Q_1(s,a)$ and $Q_2(s,a)$, $\forall s \in S$, $a \in A(s)$, arbitrarily
Initialize $Q_1(\text{terminal-state}) = Q_2(\text{terminal state}) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q_1 and Q_2 (ex. ε-greedy in $Q_1 + Q_2$)
    Take action A, observe R, S'
    With 0.5 probability:
      $Q_1(S,A) \leftarrow Q_1(S,A) + \alpha[R + \gamma Q_2(S', \arg\max_a Q_1(S',a)) - Q_1(S,A)]$
    else:
      $Q_2(S,A) \leftarrow Q_2(S,A) + \alpha[R + \gamma Q_1(S', \arg\max_a Q_2(S',a)) - Q_2(S,A)]$
    $S \leftarrow S'$;
  until S is terminal

✳ ⟨SARSA⟩ vs ⟨Q-learning⟩

⟨SARSA⟩	⟨Q-learning⟩
on-policy	off-policy
Choose an action (unnecessary for best) and update its value-function with that knowledge	choose an action and update its value-function with different action
Will converge eventually	no guarantee it will converge
slowly than learning	faster & better than SARSA

* Midterm

#1 The difference between MAB and full MDP is that in MABs, the full distribution over the results of action are ~~known~~ T/F

The main difference is that MABs have no state. MDP≠MAB, the distribution over results of an action are unknown.

#2 Exploitation is when an agent is using its current estimate of the environment and value of the current state to take the action with highest reward even if it is not globally optimal) T/F

The globally optimal action is always unknown w/o full info.

#3 To get the value function from a Q function, we need to get ~~mean~~ max of the Q value of all ~~actions~~ state at a given state. T/F

#4 A coin is thrown, ball is drawn. If the coin is head, the ball is drawn from box 1 with 3R, 2B balls. If the coin is tail, Box 2 is used which has 2R 8B balls.

a) P(red)? i) head : $\frac{1}{2} \times \frac{3}{5} = \frac{3}{10}$ ii) tail : $\frac{1}{2} \times \frac{2}{10} = \frac{1}{10}$ ∴ 0.4

b) P(black)? i) head : $\frac{1}{2} \times \frac{2}{5} = \frac{2}{10}$ ii) tail : $\frac{1}{2} \times \frac{8}{10} = \frac{4}{10}$ ∴ 0.6

c) if the red ball is drawn, what is the probability that head is drawn? $\frac{0.3}{0.4} = 0.75$

d) P(1 | red)? $\frac{P(1, \text{red})}{P(\text{red})} = \frac{0.1}{0.4} = 0.25$

#5 Jane is 80% sure that her missing notebook is in one of 2 bags, being 40% certain it is in bag1 and 40% certain it is in bag2. If a search of bag1 doesn't reveal the note book, what is the probability it is in bag 2?

$P(\text{bag2} \mid \text{not in bag1}) = \frac{P(\text{bag2, not in bag1})}{P(\text{not in bag1})} = \frac{0.4}{1-0.4} = \frac{2}{3}$

#6 Algorithm A is able to take input data D and provide a prediction of the most likely outcome of taking some action a in a particular state s.

Unfortunately, A is very unstable and doesn't converge easily. A modified algorithm B is proposed that improves stability by subtracting the current prediction (at time t) from the prediction from 3 steps earlier (at time t-3) and using the result to weight the value update. What kind of algorithm would you call B and explain.

Temporal difference method. Because algorithm B computes a difference between 2 time points to weight the value update.

#7 Let $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$. Suppose $\gamma = 0.1$ and the following sequence of rewards is received: $R_1 = 2$, $R_2 = 3$, $R_3 = 1$, $R_4 = -4$, $R_5 = 1$ and $T = 5$. Find G_1 & G_4.

$G_5 = 0$ (terminal state)

$G_4 = R_5 + \gamma G_5 = 1$

$G_3 = R_4 + \gamma G_4 = -4 + 0.1 \times 1 = -3.9$

$G_2 = R_3 + \gamma G_3 = 1 + 0.1 \times (-3.9) = 0.61$

$G_1 = R_2 + \gamma G_2 = 3 + 0.1 \times 0.61 = 3.061$

∴ $G_1 = 3.061$, $G_4 = 1$

#8 Algorithm Q-tab learning

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$ & number of episode;

Initialize $Q(s,a)$ for all $s \in S^+$, $a \in A(s)$, arbitrarily except for $Q(\text{terminal}, \cdot) = 0$;

for each episode in E do:

  repeat

    $a \leftarrow \underset{a}{\text{argmax}}\ Q(s,a)$;

  /* error : this below was actually on the exam.

    $a' \leftarrow \max_{a'} Q(s,a)$;

    take action a', observe r, s';

    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} a'(s', a') - Q(s,a)]$;

    $s \leftarrow s'$

  until s is terminal;

end

a) Is this algorithm on-policy or off-policy? explain why

on-policy. since it is using the same action from $Q(s,a)$ to obtain both the behavioral action and the update action.

b) In what situations, if any would algorithm run into problems (learning)

This algorithm could get stuck with suboptimal policies if it happens to find a feasible, but less than optimal path early on.

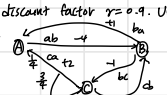
Since it does no random exploitation and always maximizes, it will often stay with the first answer it finds that completes the task.

c) Assume now that this is meant to be an implementation of Q-learning, identify errors in the implementation of the above algorithm.

a' should not be taken as the max of $Q(s,a)$. but rather the ε-greedy sample from $Q(s,a)$.

#9 Value function calculation.

Consider the following MDP with discount factor $\gamma = 0.9$. Upper case letter A, B, C : states. arc : state transitions, lower case letters ab, ba, ca, cb : actions, signed integers : rewards. Numbers : transition probabilities.



a) Consider the uniform random policy $\pi_1(s,a)$ that takes all actions from state s with equal probability. Starting with an initial value function of $V_1(A) = V_1(B) = V_1(C) = 1$, apply one synchronous iteration of iterative policy evaluation (one backup for each state) to compute a new value function $V_2(s)$

$V_2(s) = \sum_a \pi(a|s) \sum_{s',r} P(s', r \mid s, a)[r + \gamma V_1(s')]$

$V_2(A) = -4 + 0.9 \times V_1(B) = -4 + 0.9 \times 1 = -3.1$

$V_2(B) = \frac{1}{2}(1 + 0.9 \times V_1(A)) + \frac{1}{2}(1 + 0.9 \times V_1(C)) = 0.9$

$V_2(C) = \frac{1}{2}(4 + 0.9 \times V_1(B)) + \frac{1}{2}(2 + \frac{2}{3} \times 0.9 \times V_1(A) + \frac{3}{3} \times 0.9 \times V_1(C)) = 3.9$

∴ $V_2(A) = -3.1$, $V_2(B) = 0.9$, $V_2(C) = 3.9$

b) Apply one iteration of greedy policy improvement to compute a new, deterministic policy $\pi_2(s)$ assuming $V_2(A) = V_2(B) = 2$, $V_2(C) = 5$

$\pi(s) = \underset{a}{\text{argmax}}\ q_\pi(s,a)$, $q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a]$

$Q_2(B, ba) = 1 + 0.9 \times V_2(A) = 1 + 0.9 \times 2 = 2.8$

$Q_2(B, bc) = -1 + 0.9 \times V_2(C) = -1 + 0.9 \times 5 = 3.5$

$Q_2(C, ca) = 2 + 0.9 \times (\frac{1}{3} \times V_2(B) + \frac{2}{3} \times V_2(C)) = 2 + 0.9 \times (\frac{1}{3} \times 2 + \frac{2}{3} \times 5) = 5.267$

$Q_2(C, cb) = 4 + 0.9 \times V_2(B) = 4 + 0.9 \times 2 = 5.8$

∴ $\pi_2(A) = ab$, $\pi_2(B) = bc$, $\pi_2(C) = ca$

c) Apply one synchronous iteration (one backup for each state) to compute a new value function $V_2(s)$ assuming $V_1(A) = V_1(B) = V_1(C) = 2$

$V_2(A) = -4 + 0.9 \times V_1(B) = -4 + 0.9 \times 2 = -2.2$

$Q_2(B, ba) = 1 + 0.9 \times V_1(A) = 1 + 0.9 \times 2 = 2.8$

$Q_2(B, bc) = -1 + 0.9 \times V_1(C) = -1 + 0.9 \times 2 = 0.8$

$V_2(B) = \max(Q_2(B, ba), Q_2(B, bc)) = 2.8$

$Q_2(C, ca) = 2 + 0.9 \times (\frac{1}{3} \times V_1(A) + \frac{2}{3} \times V_1(C)) = 3.8$

$Q_2(C, cb) = 4 + 0.9 \times V_1(B) = 4 + 0.9 \times 2 = 5.8$

$V_2(C) = \max(Q_2(C, ca), Q_2(C, cb)) = 5.8$

∴ $V_2(A) = -2.2$, $V_2(B) = 2.8$, $V_2(C) = 5.8$

★ Assignment

#1 MAB

Consider a k-armed Bandit problem with k=4 using ε-greedy action selection, sample average action-value estimates, and initialization of $Q(a)=0$, for all actions a. Suppose the initial sequences of actions & rewards: $A_1=2$, $A_2=1$, $A_3=4$, $A_4=4$, $A_5=2$, $R_1=0$, $R_2=2$, $R_3=2$, $R_4=4$, $R_5=3$

a) List out Q-estimates for each action for each step from $t \in [1,5]$ before the action is taken. So row 1 is before any actions and row 6 is after the last action is taken

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \,||\, A_i = a}{\sum_{i=1}^{t-1} 1 \,||\, A_i = a}$$

i) $t=1$ $Q_1(a)=0$

ii) $t=2$ $Q_2(a) = \frac{R_1 \,||\, A_1=a}{1 \,||\, A_1=a}$, $A_1=2$ $\therefore Q_2(2)=0$, o.w $Q_2(a)=0$

iii) $t=3$ $Q_3(a) = \frac{R_1||A_1=a + R_2||A_2=a}{1||A_1=a \,+\, 1||A_2=a}$, $A_1=2$, $A_2=1$ $\therefore Q_3(1)=\frac{2}{1}=2$, $Q_3(2)=0$, o.w $Q_3(a)=0$

iv) $t=4$ $Q_4(a) = \frac{R_1||A_1=a + R_2||A_2=a + R_3||A_3=a}{1||A_1=a \,+\, 1||A_2=a \,+\, 1||A_3=a}$, $A_1=2$, $A_2=1$, $A_3=4$ $\therefore Q_4(1)=\frac{2}{1}=2$, $Q_4(2)=0$, $Q_4(4)=\frac{2}{1}=2$

v) $t=5$ $Q_5(a) = \frac{R_1||A_1=a + R_2||A_2=a + R_3||A_3=a + R_4||A_4=a}{1||A_1=a \,+\, 1||A_2=a \,+\, 1||A_3=a \,+\, 1||A_4=a}$, $A_1=2$, $A_2=1$, $A_3=4$, $A_4=4$ $\therefore Q_5(1)=2$, $Q_5(2)=0$, $Q_5(4)=\frac{2+4}{2}=3$

vi) $t=6$ $Q_6(a) = \frac{R_1||A_1=a + R_2||A_2=a + R_3||A_3=a + R_4||A_4=a + R_5||A_5=a}{1||A_1=a \,+\, 1||A_2=a \,+\, 1||A_3=a \,+\, 1||A_4=a \,+\, 1||A_5=a}$, $A_1=2$, $A_2=1$, $A_3=4$, $A_4=4$, $A_5=2$ $\therefore Q_6(1)=2$, $Q_6(2)=\frac{0+3}{2}=1.5$, $Q_6(4)=\frac{2+4}{2}=3$

$\therefore$ t	$a=1$	$a=2$	$a=3$	$a=4$
step 0 G_1	0	0	0	0
step 1 G_2	0	0	0	0
step 2 G_3	2	0	0	0
step 3 G_4	2	0	0	2
step 4 G_5	2	0	0	3
step 5 G_6	2	1.5	0	3

b) ε case: explore, greedy: exploit. Explain what we can tell about whether the agent definietly chose the ε case or whether it might have chosen the greedy case.

i) step 0, all Q values = 0 for all the actions. ⇒ actions will be selected random ∴ ε action.

ii) step 1, all Q values = 0 for all the actions. ∴ still ε action.

iii) $R_1 = 2$ for action 1 and other Q values are 0. So best action at this step is action 1.
But the agent selected action 4. ∴ ε action. (definitely)

iv) maximum returns: both action 1, 4. So one of it can be selected. Thus, may be ε action.

v) best action: 4, but it choose action 2 ∴ ε action (definitely)

c) Why can't we be any more sure than this?

The exploration ε case could have occured at any step, even the cases where the greedy action is chosen, it could have been randomly selected, although this unlikely.

#2 MDP

general form of the return for MDP from time step t going forward until the terminal state is reached at $t=T$: $G_t = \sum_{k=t+1}^{T} \gamma^{k-t} R_{t+k} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1} R_T$. But it also be defined recursively as for a non-terminal steps as: $G_t = R_{t+1} + \gamma G_{t+1}$. Suppose $\gamma = 0.9$, and the following sequence of rewards: $R_1=6$, $R_2=3$, $R_3=0$, $R_4=-3$, $R_5=-6$, $T=5$. Once the terminal state is reached, there are no more rewards received and the episode ends.

a) For $G_0, G_1, G_2, G_3, G_4, G_5$, write out the one step recursive form and Calculate.

$G_0 = R_1 + 0.9\, G_1 \rightarrow 6 + 0.9 \times (-3.8) = 2.578$

$G_1 = R_2 + 0.9\, G_2 \rightarrow 3 + 0.9 \times (-7.56) = -3.8$

$G_2 = R_3 + 0.9\, G_3 \rightarrow 0 + 0.9 \times (-8.4) = -7.56$

$G_3 = R_4 + 0.9\, G_4 \rightarrow -3 + 0.9\,(-6) = -8.4$

$G_4 = R_5 + 0.9\, G_5 \rightarrow R_5 = -6$

$G_5 = R_6 + 0.9\, G_6 \rightarrow 0$ (terminal state)

once terminal state reached, no reward ∴ $G_6 = G_5$

$\therefore G_6 = 2.578$, $G_1 = -3.8$, $G_2 = -7.56$, $G_3 = -8.4$, $G_4 = -6$, $G_5 = 0$

b) calculate G_0 by non-recursive formula.

$G_0 = R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \gamma^4 R_5$

$= 6 + (0.9)(3) + (0.9)^2 \times 0 + (0.9)^3 \times (-3) + (0.9)^4 \times (-6)$

$= 2.578$

#3 Policy evaluation

Figure shows a rectangular grid world representation of a simple finite MDP. The cells of the grid correspond to the states of the environment. At each cell, 4 actions are possible: north, south, east, west. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward -1. Other actions result in a reward of 0. With a discount factor $\gamma = 0.9$, and equal probability all 4 actions.

Specify Bellman equation for the setting using the given equiprobable, random policy.

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.4
0.1	0.7	0.7	0.4	-0.4
-1	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2

a) Write down Bellman equation and describe what is known about each component in general terms.

$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\,[r + \gamma V_\pi(s')]$

For the given grid world, all actions have equal probability, so $\pi(a|s) = \frac{1}{4}$ for all actions, $\gamma = 0.9$.

b) Show the calculation for state J

i) if it goes up: $\frac{1}{4}(0 + 0.9 \times 5.3) = 1.1925$

ii) if it goes left: $\frac{1}{4}(0 + 0.9 \times 2.3) = 0.5175$

iii) if it goes right: $\frac{1}{4}(0 + 0.9 \times 0.5) = 0.1125$

iv) if it goes down: $\frac{1}{4}(0 + 0.9 \times 0.4) = 0.09$

$\therefore V_\pi(J) = 1.1925 + 0.5175 + 0.1125 + 0.09 = 1.9125$

c) Show the calculation for state K

i) if it goes up: $\frac{1}{4}(0 + 0.9 \times (-1.2)) = -0.27$

ii) if it goes left: $\frac{1}{4}(0 + 0.9 \times (-1.4)) = -0.3125$

iii) if it goes right: $\frac{1}{4}(-1 + 0.9 \times V_\pi(k))$

iv) if it goes down: $\frac{1}{4}(-1 + 0.9 \times V_\pi(k))$

$\therefore V_\pi(K) = -0.27 - 0.3125 - 0.5 + 0.45\, V_\pi(k)$

$V_\pi(k) = 1.99$