What is the main difference between passive reinforcement learning and evaluating a policy π by solving an MDP (Markov Decision Process) model?

A

There is no difference because passive reinforcement learning is also to evaluate a policy ﻿π﻿.

B

Passive reinforcement learning can find the optimal policy.

C

In passive reinforcement learning, the transition model and the reward function are unknown.

D

Passive reinforcement learning can learn the action-value function Q(s,a) while Policy Evaluation with MDP can learn the state-value function V(s).

@@@@@@@@@@@@@

Initialize:

V(s) ER, arbitrarily, for all s€ S

Returns(s) ← an empty list, for all s Є S

Loop forever (for each episode):

Generate an episode following π: So, Ao, R1, S1, A1, R2,..., ST-1, AT-1, RT

G←0

Loop for each step of episode, t = T-1, T-2, ..., 0:

G← YG + Rt+1

Unless St appears in So, S1,..., St-1:

Append G to Returns (St)

V(St) average (Returns (St))

What does the following algorithm do?

A

Estimating a state-value function for a policy π.

B

Estimating an action-value function for a policy π.

C

Estimating the optimal state-value function.

D

Estimating the optimal action-value function.

@@@@@@@@@@@@@@

Algorithm parameter: step size a € (0,1]

Initialize V(s), for all s € S†, arbitrarily except that V (terminal) = 0

Loop for each episode:

Initialize S

Loop for each step of episode:

Aaction given by π for S

Take action A, observe R, S'

V (S) ← V (S) + a[R + 7V (S′) − V (S)]

S← S'

until S is terminal

What is the following algorithm?

A

A temporal difference learning algorithm to estimate the state-value function for a policy π.

B

A Monte Carlo learning algorithm to estimate the state-value function for a policy π.

C

SARSA algorithm for estimating the optimal state-value function.

D

SARSA algorithm for estimating the optimal action-value function.

@@@@@@@@@

A screenshot of a graph

Description automatically generated

Given the following environment with five states A, B, C, D, and E where A and D are two terminal states.

Let's consider the following policy π: π(A)=exit, π(B)=right, π(C)=right, π(D)=exit, π(E)=up.

We use the **Temporal Difference Learning** method to evaluate the policy π.

We have two transitions as in the below picture. (B,right,C,-4) gives us a reward of -4. (C, right, D,-2) gives us a reward of -2.

Assuming the discount factor gamma=1, learning rate alpha=1/2, **compute the expected utility of the five states after the two transitions V(A)=?, V(B)=?, V(C)=?, V(D)=?, V(E)=?**

**A**

V(A)=0, V(B)=-2, V(C)=2, V(D)=6, V(E)=0

**B**

V(A)=0, V(B)=-2, V(C)=-2, V(D)=3, V(E)=0

**C**

V(A)=-2, V(B)=2, V(C)=2, V(D)=6, V(E)=0

**D**

V(A)=0, V(B)=2, V(C)=-2, V(D)=6, V(E)=0

@@@@@@@@@@@@@@@@@

What is the key difference between **passive** reinforcement learning and **active** reinforcement learning?

**A**

Passive RL is to learn the value function for a fixed policy *π* while active RL is to learn the optimal policy/value function.

**B**

The transition function *P*(*s*′∣*s*,*a*) is unknown in active RL.

**C**

The reward function *R*(*s*,*a*,*s*′) is unknown in active RL.

**D**

We can employ the Value Iteration/Policy Iteration algorithms to solve active RL

@@@@@@@@@@@@@@@@@@@@

A white background with black text

Description automatically generated

How should we change the following Reinforcement Learning algorithm so that we can estimate an optimal value function?

**A**

Estimating the action-value function *Q*(*St*​,*At*​) instead of the state-value function *V*(*St*​).

**B**

Performing one-step lookahead to improve the current estimated state-value function *V*(*St*​)←*a*max​*P*(*St*+1​∣*St*​,*At*​)[*R*(*St*​,*At*​,*St*+1​)+*γV*(*St*+1​)]

**C**

Performing policy improvement with respect to the current estimated action-value function, i.e., *π*(*St*​)←arg*a*max​*Q*(*St*​,*a*).

**D**

Choose *S*0​∈*S*, and *A*0​∈A(*S*0​) randomly such that all pairs have probability >0 .

**E**

A, B, and C need to be done.

**F**

A, C, and D need to be done.

**G**

B and D need to be done.

**H**

A, B, C, and D need to be done.

@@@@@@@@@@@@@@@@@@@

What is the *Q*(*s*,*a*) update rule in Q-Learning? *α* is the learning rate, *γ* is the discount factor.

**A**

﻿*Q*(*s*,*a*)←(1−*α*)*Q*(*s*,*a*)+*α*(*r*+*γQ*(*s*′,*a*′))﻿

**B**

﻿*Q*(*s*,*a*)←(1−*α*)*Q*(*s*,*a*)+*α*(*r*+*γa*′max​*Q*(*s*′,*a*′))﻿

**C**

﻿﻿*Q*(*s*,*a*)←*α*(*r*+*γa*′max​*Q*(*s*′,*a*′)−*Q*(*s*,*a*))

**D**

*Q*(*s*,*a*)←*Q*(*s*,*a*)+*α*(*r*+*γQ*(*s*′,*a*′)−*Q*(*s*,*a*))

**E**

All are correct.

@@@@@@@@@@@

A is the set of actions a's we can perform at each state *s*∈*S*. Let *n*=∣*A*∣ be the number of actions.

An ﻿*ε*﻿-greedy policy would choose a random action with probability ﻿*ε*﻿.

Regarding an *ε*-greedy policy with respect to a action-value function *Q*(*s*,*a*), in each step of **Q-Learning with**﻿*ε*﻿**-greedy exploration**, what is the probability that the action with the maximum *Q*(*s*,*a*) value would be sampled?

**A**

*nε*​

**B**

﻿*ε*﻿

**C**

1−*ε*

**D**

1−*ε*+*nε*​  
@@@@@@@@@@@@@@@@@@@@@@

A white rectangular object with black text

Description automatically generated

We are using **Approximate Q-Learning** with linear Q-value approximation for a basic Pacman game with two features: the reciprocal distance to the nearest food dot *fDOT*​ (with weight *wDOT*​)and the reciprocal distance to the nearest ghost *fGST*​(with weight *wGST*​).

Let's consider the current state s and action NORTH (going up) in the picture below. After Pacman performs the action NORTH, the blue ghost attacks Pacman, the obtained reward is -1000 in this transition, and the game ends.

**Question: Using this transition, how the weights***wDOT*​ **and** *wGST*​ **are updated?**

Note: discount factor gamma *γ*=1, and learning rate alpha *α*=0.001.

**A**

*wDOT*​≈−3.5,*wGST*​≈2

**B**

*wDOT*​≈3.5,*wGST*​≈−2

**C**

﻿﻿*wDOT*​≈3.5,*wGST*​≈−10

**D**

*wDOT*​≈−3.5,*wGST*​≈10