Subject:
Year. Month. Date. ( )

$V_2(2,2) = \max($ ۱- بیاورد برای خروج بیاورد حساب میکنیم

$V_1(2,2) = \max(0.8(0 + 0.9\times 5) + 0.1\times(0+0) + 0.1(0+0),$
$0, 0.1\times(0.9\times 5), (0.1\times 0.9\times 0.5)) = 3.6$

بالا، پایین، چپ، راست

$V_1(1,2) = \max(0.1(0.9\times -0.5), -0.5, 0, 0.8\times 0.9\times -0.5) = 0$
-0.36

$V_1(1,1) = \max(0,0,0,0) = 0$

$V_1(2,1) = (0,0,0,0) = 0$

$V_2(2,2) = 0.36 \quad V(2,1)$

$V_2(2,1) = \max(0.8\times 0.9\times 3.6, 0.1\times 0.9\times 0.36) = 2.592$

$V_2(1,2) = \max(0.8\times 0.9\times 3.6 = 0.1\times 0.9\times 5, 0+0.1\times 2.592, -0.5, -)$
بالا 2.1 چپ پایین راست <0

$= 2.1$

$V_2(1,1) = 0$

	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)
V_0	0	0	-5	0	0	5
V_1	0	0	-5	0	3.6	5
V_2	0	2.1 ✱2	-5	2.592	3.6	5

	1,1	1,2	1,3	2,1	2,2	2,3
$\pi^*(s)$ =	up	up	—	Right	Right	

Subject:
Year. Month. Date. ( )

$1167 = \frac{5+5-5}{3} = (1,1)$ $\frac{5+9}{2} = 5$ $\in (2,2)$ ③

running 1-3+ePTD ④

$V(S_t) = V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ (0.9)

$V(1,1) = 0 + 0.1(0 + 0.9 \times 0 - 0) = 0$

$V(1,2) = 0.1 \times (0 + 0.9 \times .05 + 0) = .0145$

$V(1,1) = 0.1 \times -0.145 = -0.0145$

$V(1,2) = -0.145 + 0.1(0) = -0.145$

$V(2,2) = 0 + 0.145 = 0.145$

$V(1,1) = -0.145 * 0.1 \times 0.95 = -0.0145$

$V(2,1) = 0.0945$

$V(2,2) = 0.145 + 0.145 = 0.19$

run it all again

$V(1,1) = -0.09$

$V(1,2) = -0.19$

$V(1,1) = -0.118$ $V(1,2) = -0.181$

$V(2,2) = 1135$

$V(1,1) = -0.19 + 0.1 \times 0.19 \times 0.95 = \sim$

$V(2,1) = 0.09$

PAPCO

$V(2,2) = 118$

Deep Q-learning

Deep Q-Learning (DQL) is an advanced form of Q-Learning, a reinforcement learning algorithm. It combines Q-Learning with deep neural networks to create a system that can learn optimal policies for decision-making problems by interacting with an environment. DQL uses a neural network as a function approximator to predict the quality (Q-value) of actions given different states, allowing it to handle high-dimensional input spaces that traditional Q-Learning cannot.

Deep Q-Learning works by using a neural network, often called a Q-network, to approximate the Q-value function. Here's a simplified overview of the process:

Initialize:

Start with a random policy and an empty experience replay buffer.

Observe:

The agent interacts with the environment to obtain state, action, reward, and next state information.

Store:

Save these experiences in the replay buffer.

Sample:

Randomly sample a batch of experiences from the buffer.

Learn:

Use these samples to update the Q-network by minimizing the loss between predicted Q-values and target Q-values (calculated using the Bellman equation).

Repeat:

Continue interacting with the environment and updating the network.

The use of experience re-play and fixed Q-targets helps stabilize training by reducing correlations between samples and keeping target values consistent for short periods.