RL Pong Game

Dr.mohammadi

Sara Younesi – Narges Mashayekhi

فهرست

۱. پیاده سازی بازی با QL

۲. الگوریتم QL

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
  until S is terminal

QLپیاده سازی

```
for _ in range(10000000000):
    action = None
    r = 0
    for a in ACTIONS:
        if (state, a) in Q and r < Q[(state, a)]:
            r = Q[(state, a)]
            action = a
    if action is None or random.random() <= epsilon:
        action = env.action_space.sample()
    N, R, T, Tc, data = env.step(action)
    N = Func2(N)

    if not (state, action) in Q:
        Q[(state, action)] = 0.0

    if Q[((state, action))] != 0.0:
        print("action for  train  ", action,
              "  State , Action", Q[((state, action))])

    Next_Next = 0
    for a in ACTIONS:
        if (N, action) in Q:
            Next_Next = max(Next_Next, Q[(N, action)])

    Q[(state, action)] = (1 - alpha) * Q[(state, action)] + \
        alpha * (R + gamma * Next_Next)

    if epsilon > last_epsilon:
        epsilon -= 0.001
    state = N
    if T or Tc:
        N, data = env.reset()
SaveQToFile
env.close()
```

همگرا شدن به یک عدد در نهایت اما در تعدا استیت زیاد غیر بهینه

C:\Windows\system32\cmd.exe

```
Action:  0   Q(state, action) -0.09909284113834355
Action:  0   Q(state, action) -0.07927427291067485
Action:  5   Q(state, action) -0.0018447668222170402
Action:  3   Q(state, action) -0.00048358380812121913
Action:  3   Q(state, action) -0.0003868670464969753
Action:  3   Q(state, action) -0.0003094936371975803
Action:  0   Q(state, action) -0.06341941832853988
Action:  3   Q(state, action) -0.00024759490975806425
Action:  3   Q(state, action) -0.0001980759278064514
Action:  2   Q(state, action) -9.331719504571945e-10
Action:  5   Q(state, action) -0.0014758134577736322
Action:  3   Q(state, action) -0.00015846074224516114
Action:  4   Q(state, action) -0.20094447329659976
Action:  3   Q(state, action) -0.00012676859379612892
Action:  1   Q(state, action) -0.00021176204786471214
Action:  1   Q(state, action) -0.00016940963829176973
Action:  0   Q(state, action) -0.050735534662831906
Action:  0   Q(state, action) -0.040588427730265525
Action:  3   Q(state, action) -0.00010141487503690314
Action:  2   Q(state, action) -7.465375603657557e-10
Action:  2   Q(state, action) -5.972300482926045e-10
Action:  4   Q(state, action) -0.16075557863727982
Action:  0   Q(state, action) -0.03247074218421242
Action:  5   Q(state, action) -0.001180650766218906
Action:  0   Q(state, action) -0.025976593747369936
Action:  5   Q(state, action) -0.0009445206129751248
Action:  4   Q(state, action) -0.12860446290982386
Action:  1   Q(state, action) -0.00013552771063341578
Action:  0   Q(state, action) -0.02078127499789595
```

۲. پیاده سازی با approximate

Compatible Function Approximation

Theorem (Compatible Function Approximation Theorem)

If the following two conditions are satisfied:

1. *Value function approximator is compatible to the policy*

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(s,a)$$

2. *Value function parameters w minimise the mean-squared error*

$$\varepsilon = \mathbb{E}_{\pi_\theta}\left[\left(Q^{\pi_\theta}(s,a) - Q_w(s,a)\right)^2\right]$$

Then the policy gradient is exact,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)\right]$$

۲. ویژگی اول

```python

def Ball(state):
    for i in range(34, 194):  # amoodi
        for j in range(0, 160):  # ofoghi
            if state[i][j][0] == 236 and state[i][j][1] == 236 and state[i][j][2] == 236:
                return i, j
    return 0, 0
# Ball


def NextStateOfAgent(state, action):
    dy = 0
    if action == 3:
        dy = -10
    if action == 2:
        dy = 10
    for i in range(34, 194):  # amoodi
        for j in range(140, 144):  # ofoghi
            if state[i][j][0] == 92 and state[i][j][1] == 186 and state[i][j][2] == 92:
                return i + dy + 8, j

    return 0, 0
# PlatePos

```

۲. ویژگی دوم

```python

def Feature3(state):
    counter = 0
    for i in range(34, 193):
        for j in range(138, 141):
            if state[i][j][0] == 236 and state[i][j][1] == 236 and state[i][j][2] == 236 and state[i][j+1][0] == 92 and state[i][j+1][1] == 186 an
                counter += 1
    return counter
# fiercount

```

۲. ویژگی سوم

```python
def Feature4(state):
    counter = 0
    for i in range(34, 194):
        for j in range(0, 16):
            if state[i][j][0] == 236 and state[i][j][1] == 236 and state[i][j][2] == 236:
                counter += 1
    return counter
```

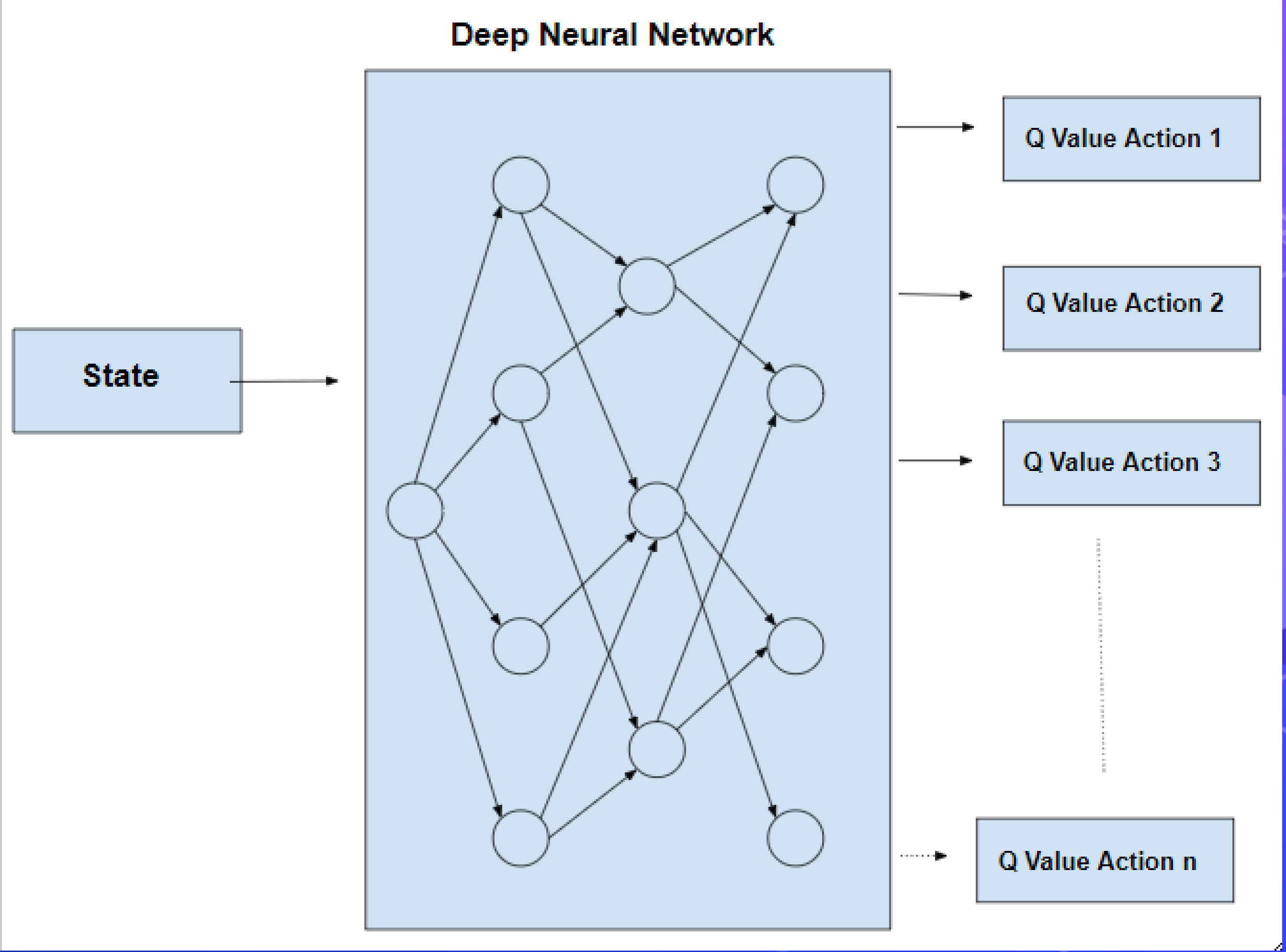
۲. ویژگی پیشنهادی

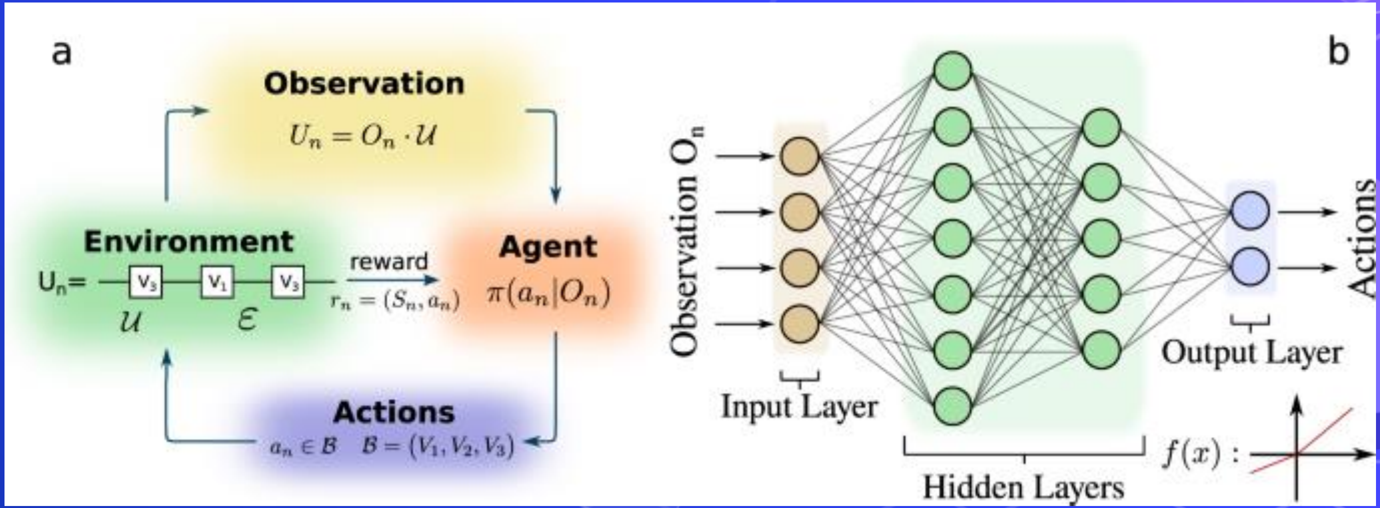
```
def Feature4(state):
    counter = 0
    for i in range(34, 194):
        for j in range(0, 16):
            if state[i][j][0] == 236 and
state[i][j][1] == 236 and state[i][j][2]
== 236:
                counter += 1
    return counter


def Feature5(state):
    x1, y1 = Ball(state)
    x2, y2 = PlateEnemyPos(state)
    return abs(x2-x1)/100, abs(y2-y1)/100
# BallPlateDistanceToEnemy
```

۲. الگوریتم پیاده سازی شده

```python
env = gym.make("ALE/Pong", render_mode="human")
state, info = env.reset(seed=5)
WEIGHTS["W1"], WEIGHTS["W2"], WEIGHTS["W3"] = ReadQFromFile()

for i in range(10000):
    action = None
    optimizedValue = math.inf
    for optimizedAction in ACTIONS:
        value = updateWeight(state, optimizedAction)
        if optimizedValue > value:
            optimizedValue = value
            action = optimizedAction

    if action is None or random.random() <= epsilon:
        action = env.action_space.sample()
    next_state, reward, terminated, truncated, info = env.step(action)
    futureReward = 0
    for a in ACTIONS:
        print(a)
        futureReward = max(futureReward, updateWeight(next_state, a))
    difference = (reward + gamma * futureReward) - updateWeight(state, action)
    dx, dy = BallPlateDistanceToAgent(state, action)
    WEIGHTS["W1"] = round(WEIGHTS["W1"] + alpha * difference * dx, 6)
    WEIGHTS["W2"] = round(WEIGHTS["W2"] + alpha * difference * dy, 6)
    WEIGHTS["W3"] = round(WEIGHTS["W3"] + alpha *
                          difference * Feature3(state), 6)
    print(WEIGHTS["W1"],WEIGHTS["W2"],WEIGHTS["W3"],WEIGHTS["W4"] )
    state = next_state
    if terminated or truncated:
        next_state, info = env.reset()
SaveQToFile()

env.close()
```

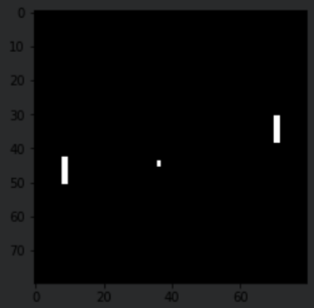
مقایسه با نحوه ی پیاده سازی
شبکه عصبی و یادگیری عمیق


Deep Neural Network
State
Q Value Action 1
Q Value Action 2
Q Value Action 3
Q Value Action n



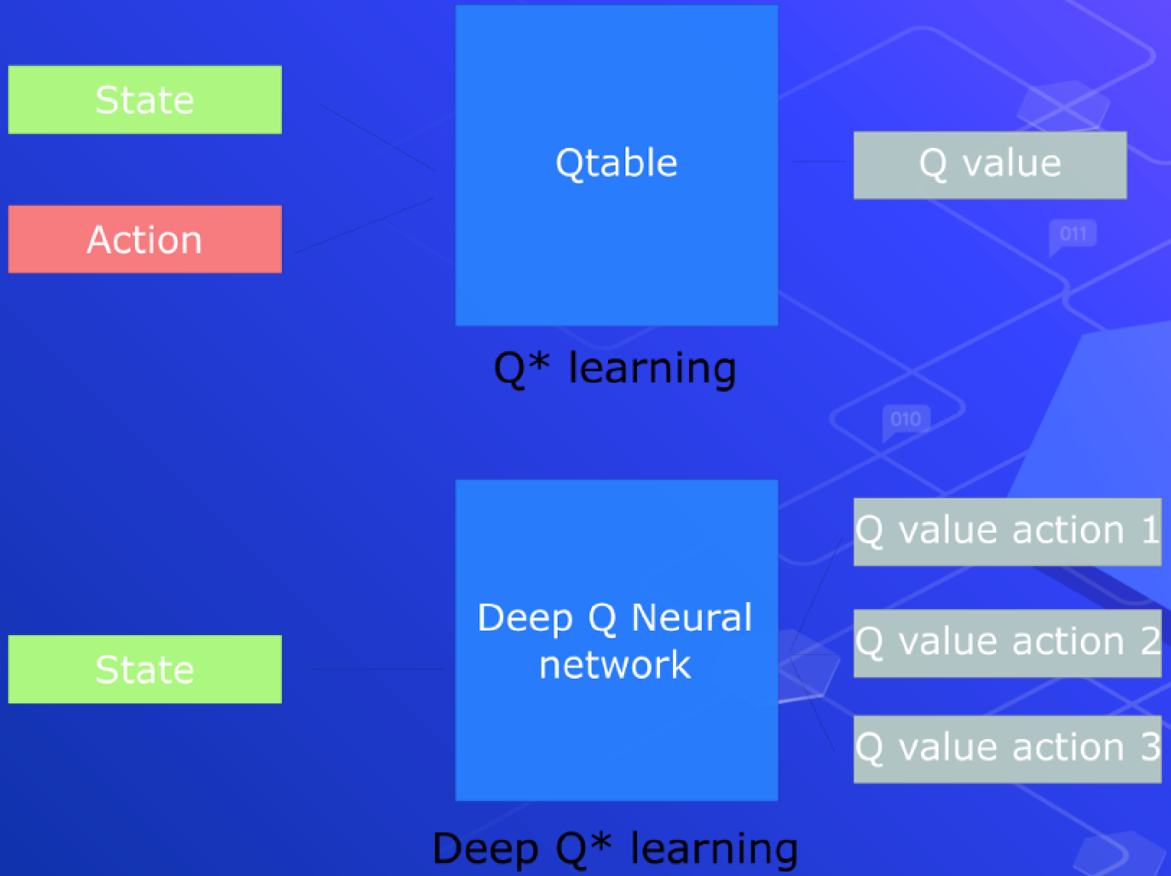
a
Observation
$U_n = O_n \cdot \mathcal{U}$
Environment
$U_n =$ V3 V1 V3
$\mathcal{U}$
$\mathcal{E}$
reward
$r_n = (S_n, a_n)$
Agent
$\pi(a_n|O_n)$
Actions
$a_n \in \mathcal{B}$ $\mathcal{B} = (V_1, V_2, V_3)$
b
Observation O_n
Input Layer
Hidden Layers
Output Layer
Actions
$f(x)$:

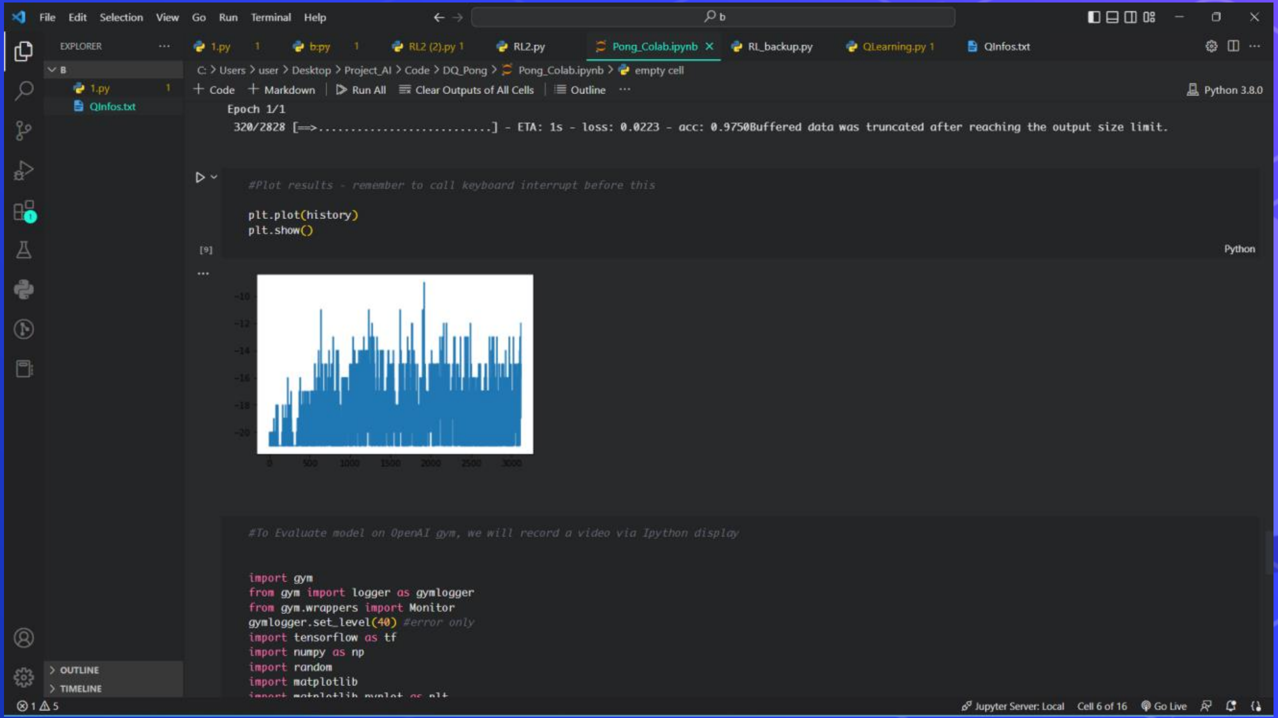
```
#Show preprocessed

obs_preprocessed = prepro(observation).reshape(80,80)
plt.imshow(obs_preprocessed, cmap='gray')
plt.show()
```

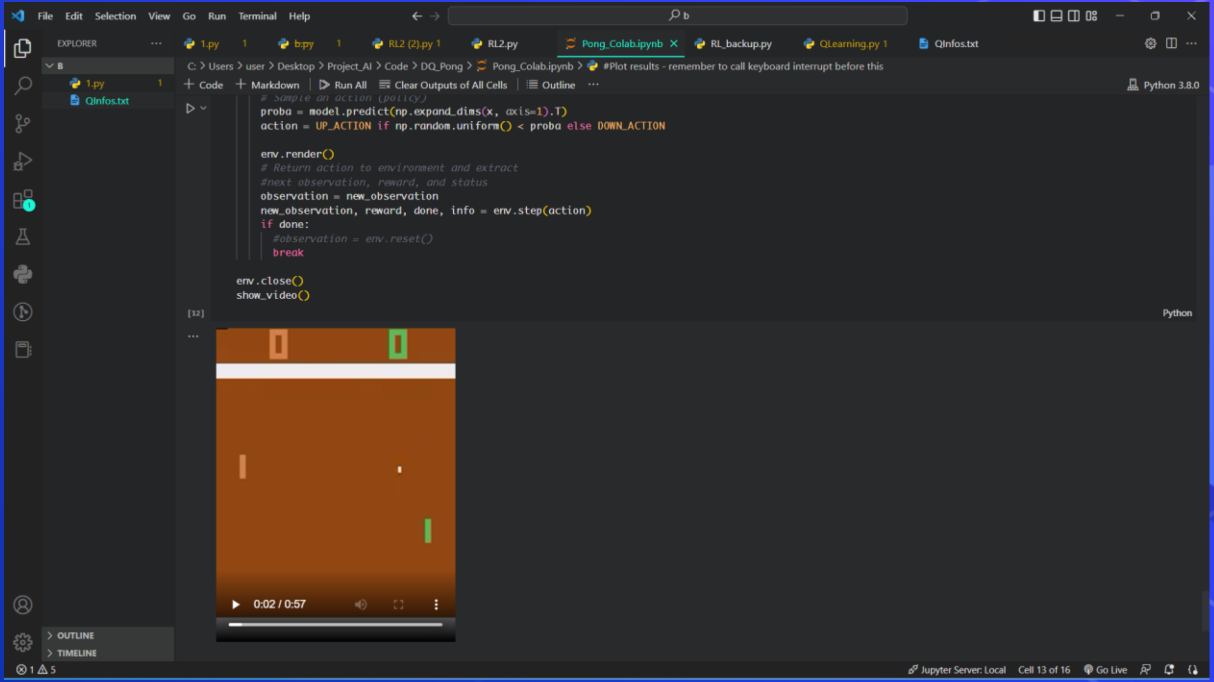


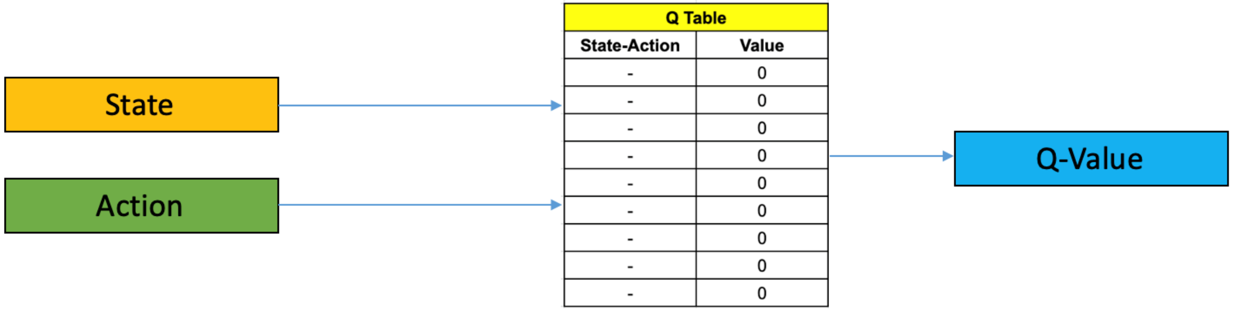
Jupyter Server: Local   Cell 5 of 16

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/nn_impl.py:183: where (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 200)               1280200
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 201
=================================================================
Total params: 1,280,401
Trainable params: 1,280,401
Non-trainable params: 0
_________________________________________________________________
None

Using TensorFlow backend.

"\ninputs = keras.layers.Input(shape=(80,80))\nchanneled_input = keras.layers.Reshape((80,80,1))(inputs) # Conv2D requries (batch, height, width, channels)  so
we need to create a dummy channel \nconv_1 = keras.layers.Conv2D(filters=10,kernel_size=20,padding='valid',activation='relu',strides=(4,4),use_bias=False)
(channeled_input)\nconv_2 = keras.layers.Conv2D(filters=20,kernel_size=10,padding='valid',activation='relu',strides=(2,2),use_bias=False)(conv_1)\nconv_3 =
keras.layers.Conv2D(filters=40,kernel_size=3,padding='valid',activation='relu',use_bias=False)(conv_2)\nflattened_layer = keras.layers.Flatten()
(conv_3)\nsigmoid_output = keras.layers.Dense(1,activation='sigmoid',use_bias=False)(flattened_layer)\nmodel =
keras.models.Model(inputs=inputs,outputs=sigmoid_output)\nmodel.compile(loss='binary_crossentropy', optimizer='adam', metrics=
['accuracy'])\nprint(model.summary())\n"
```


State
Action
Qtable
Q value
Q* learning
State
Deep Q Neural network
Q value action 1
Q value action 2
Q value action 3
Deep Q* learning

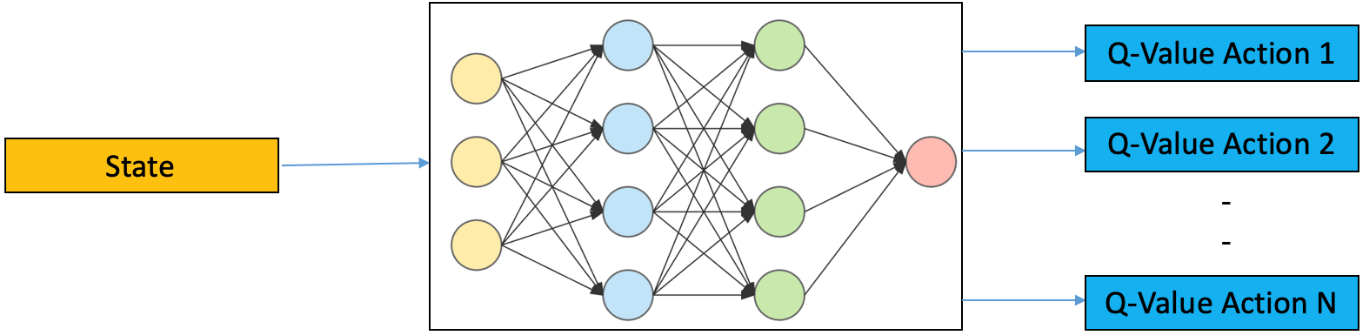

File Edit Selection View Go Run Terminal Help
EXPLORER
1.py
b.py
RL2 (2).py
RL2.py
Pong_Colab.ipynb
RL_backup.py
QLearning.py
QInfos.txt
C: > Users > user > Desktop > Project_AI > Code > DQ_Pong > Pong_Colab.ipynb > empty cell
+ Code + Markdown Run All Clear Outputs of All Cells Outline
Python 3.8.0
Epoch 1/1
320/2828 [==>..........................] - ETA: 1s - loss: 0.0223 - acc: 0.9750Buffered data was truncated after reaching the output size limit.
#Plot results - remember to call keyboard interrupt before this
plt.plot(history)
plt.show()
[9]
Python
#To Evaluate model on OpenAI gym, we will record a video via Ipython display
import gym
from gym import logger as gymlogger
from gym.wrappers import Monitor
gymlogger.set_level(40) #error only
import tensorflow as tf
import numpy as np
import random
import matplotlib
OUTLINE
TIMELINE
Jupyter Server: Local
Cell 6 of 16
Go Live


File Edit Selection View Go Run Terminal Help
EXPLORER
1.py
b.py
RL2 (2).py
RL2.py
Pong_Colab.ipynb
RL_backup.py
QLearning.py
QInfos.txt
C: > Users > user > Desktop > Project_AI > Code > DQ_Pong > Pong_Colab.ipynb > #Plot results - remember to call keyboard interrupt before this
+ Code + Markdown Run All Clear Outputs of All Cells Outline
Python 3.8.0
proba = model.predict(np.expand_dims(x, axis=1).T)
action = UP_ACTION if np.random.uniform() < proba else DOWN_ACTION
env.render()
# Return action to environment and extract
#next observation, reward, and status
observation = new_observation
new_observation, reward, done, info = env.step(action)
if done:
#observation = env.reset()
break
env.close()
show_video()
[12]
Python
0:02 / 0:57
OUTLINE
TIMELINE
Jupyter Server: Local
Cell 13 of 16
Go Live



Q Table	
State-Action	Value
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0

Q Learning



Deep Q Learning

منابع

OpenAI

What is Reinforcement Learning? A Comprehensive Overview (techtarget.com)

Reinforcement Q-Learning from Scratch in Python with OpenAI Gym – LearnDataSci

Deep Reinforcement Learning: Pong from Pixels (karpathy.github.io)

numpy-tutorials/tutorial-deep-reinforcement-learning-with-pong-from-pixels.md at main · numpy/numpy-tutorials (github.com)

با سپاس از همراهی و توجه شما

پایان