درس : مبانی و کاربردهای هوش مصنوعی

استاد: جوانمردی

دانشجو: مهدی منصوری خواه

شماره دانشجویی: 9931056

گزارش پروژه – سوم(پکمن)

1- تکرار ارزش

runValueIteration

با استفاده از این تابع به تعداد تکراری که برایمان مشخص شده است عملیات بروزرسانی مقدار ارزش را بدست می اوریم .پیاده سازی آن هم به این صورت است که به ازای تمامی حالت هایی که پایانه نباشند، بهترین Q را بدست آورده و برابر با مقدار جدید vآن حالت می گذاریم.

computeQValueFromValues

در این تابع، Q_value مربوط به action در state کنون را از روی value ها محاسبه کرده ایم. سپس با استفاده از این تابع و فرمولی که در لکچر های استاد آورده شده بود، میتوانیم مقدار value Qرا حساب کنیم ابتدا روی تمام state های ثانویه لوپ زده و مقدار Q_value را با حاصل T *(R+&*V) جمع می کنیم که& همان تخفیف است و R همان سود است و V هم همان value (s’)می باشد.

computeActionFromValues

و این عمل به جهت پیدا کردن حرکت مورد نظر از روی value به دست آمده می باشد.: در این تابع یکargmax گرفته شده است و پیاده سازی این قسمت مانند قسمت قبل می باشد.

```python
def runValueIteration(self):
    # Write value iteration code here
    "*** YOUR CODE HERE ***"
    for i in range(self.iterations):
        counter = util.Counter()
        for state in self.mdp.getStates():
            if not self.mdp.isTerminal(state):
                counter[state] = max(
                    [self.computeQValueFromValues(state, action) for action in self.mdp.getPossibleActions(state)])
        self.values = counter
```

```python
def computeQValueFromValues(self, state, action):
    """
      Compute the Q-value of action in state from the
      value function stored in self.values.
    """
    "*** YOUR CODE HERE ***"
    value = 0.
    Lambda = self.discount
    for sPrime, T in self.mdp.getTransitionStatesAndProbs(state, action):
        R = self.mdp.getReward(state, action, sPrime)
        value += T * (R + Lambda * self.values[sPrime])
    return value
    # util.raiseNotDefined()
```

```
"*** YOUR CODE HERE ***"
if self.mdp.isTerminal(state):
    return None
actions = self.mdp.getPossibleActions(state)
values = [self.computeQValueFromValues(state, action) for action in actions]
maxIndex = values.index(max(values))
return actions[maxIndex]
util.raiseNotDefined()
```

2- تجزیه وتحلیل عبور از پل

answerDiscount = 0.9

answerNoise = 0.003

مقدار نویز را کم کردم تا بدین وسیله عامل با اطمینان بیشتری بتواند از خانه هایی که امتیاز -100 به عامل میدهد فرار کند.

3- سیاست ها

خروجی نزدیک را ترجیح دهید ،ریسک صخره را بپذیرید.

answerDiscount = 0.1
answerNoise = 0.0
answerLivingReward =-0.1

برای اینکه به سمت خروجی نزدیک برویم باید مقدار تخفیف کم باشد همچنین وقتی ریسک صخره هارا انجام میدهد که نویز کمتری داشته باشد.

خروجی نزدیک را ترجیح دهید اما از صخره را اجتناب کنید.

answerDiscount = 0.5
answerNoise = 0.4
answerLivingReward =-0.7

برا اینکه ریسک صخره را نکند باید مقدار نویز بیشتر از قبل باشد .
همچنین باید مقدار تخفیف و جریمه هر حرکت را بیشتر کنیم که آن را با آزمایش و خطا آنها را به ترتیب 4.0 و -7.0 بدست آوردم

خروجی دور را ترجیح دهید خطر صخره را ریسک کنید.

answerDiscount = 1.0
answerNoise = 0.0
answerLivingReward =-0.1

این همان حالت اول است با این تفاوت که به جای رفتن به سمت خروجی نزدیک باید با افزایش تخفیف آنرا به سمت خروجی دورتر بکشانیم.

خروجی دور را ترجیح دهید خطر صخره را اجتناب کنید.

answerDiscount = 1.0
answerNoise = 0.4

answerLivingReward =-0.1

این همان حالت دوم است با این تفاوت که به جای رفتن به سمت خروجی نزدیک باید با افزایش تخفیف آنرا به سمت خروجی دورتر بکشانیم.

از هر دو خروجی و صخره اجتناب کنید بنابراین اجرای یک قسمت هرگز نباید پایان یابد.

answerDiscount = 1.0

answerNoise = 0.0

answerLivingReward = 17.0

برا اینکه از خروجی ها و صخره دور شود، مقدار پاداش به ازای هر حرکت را مقدار مثبت بزرگی گذاشتم تا عامل به جای اینکه بخواهد خود را به خروجی ها برساند مکررا در نقشه حرکت کند و امتیاز +17 برای هر جابجایی بگیرد

```python
def question3a():
    answerDiscount = 0.1
    answerNoise = 0.0
    answerLivingReward = -0.1
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question3b():
    answerDiscount = 0.5
    answerNoise = 0.4
    answerLivingReward = -0.7
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question3c():
    answerDiscount = 1.0
    answerNoise = 0.0
    answerLivingReward = -0.1
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question3d():
    answerDiscount = 1.0
    answerNoise = 0.4
    answerLivingReward = -0.1
    return answerDiscount, answerNoise, answerLivingReward
```

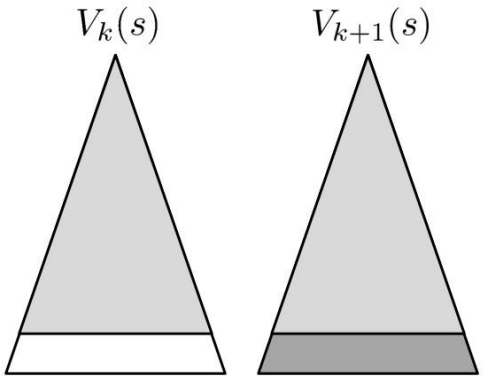
```python
def question3e():
    answerDiscount = 1.0
    answerNoise = 0.0
    answerLivingReward = 17.0
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'
```

گزارش:آیا استفاده از الگوریتم تکرار ارزش تحت هر شرایطی به همگرایی می انجامد؟

با توجه به اسلایدهای درس داریم:

Convergence*

- How do we know the V_k vectors are going to converge?
- Case 1: If the tree has maximum depth M, then V_M holds the actual untruncated values
- Case 2: If the discount is less than 1
  - Sketch: For any state V_k and V_{k+1} can be viewed as depth k+1 expectimax results in nearly identical search trees
  - The difference is that on the bottom layer, V_{k+1} has actual rewards while V_k has zeros
  - That last layer is at best all R_{MAX}
  - It is at worst R_{MIN}
  - But everything is discounted by γ^k that far out
  - So V_k and V_{k+1} are at most γ^k max|R| different
  - So as k increases, the values converge



بله همگراه می شود.

4- تکرار ارزش ناهمزمان

در این قسمت قرار است به جای اینکه مقدار های ارزش را به صورت خوشه ای آپدیت کنیم، جدا جدا آپدیت کنیم و همچنین باید توجه کرد که افزایش تکرار بعد از هر مرتبه گردش درکل استیت ها صورت نمی گیرد بلکه در هر استیت صورت می گیرد.

```python
def runValueIteration(self):

    "*** YOUR CODE HERE ***"
    k = 0
    iteration = self.iterations
    while k < iteration:
        for state in self.mdp.getStates():
            value = util.Counter()
            actions = self.mdp.getPossibleActions(state)
            for action in actions:
                value[action] = self.computeQValueFromValues(state, action)
            self.values[state] = value[value.argMax()]
            k += 1
            if k == iteration:
                return None
```

سوال: روشهای بروزرسانیای کهدر بخش اول(بروزرسانی با استفاده ازbatch) ودر این بخش(بروزرسانی به صورت تکی) پیاده کرده اید را با یکدیگر مقایسه کنید. (یک نکته مثبت و یک نکته منفی برای هرکدام)

مزیت batch نسبت به مدل تکی این است که به تعداد کمتری iteration برای همگرا شدن نیاز دارد به دلیل اینکه در هر زمان همه استیت ها را آپدیت میکند. ولی مزیت مدل تکی نسبت به batch اینکه نیاز به حافظه کمتری دارد به دلیل اینکه در هر زمان فقط یک حالت را آپدیت میکند.

5- تکرار ارزش اولویت بندی شده

در این الگوریتم برای کاهش محاسبات، اولویت بروزرسانی مقدار هر value ابتدا با استیت هایی است که از آنها به استیت فعلی میتوانیم برویم(predecessors)بدین منظور در گام هایی که در دستورکار گفته شده است، میتوانیم اولویت را تعیین کنیم و بر مبنای اولویت بروزرسانی را انجام دهیم

.پیاده سازی این قسمت دقیقا معادل آن چیزی است که در دستورکار توضیح داده شده است .برای این بخش کامنت مناسب به ازای هر گام گذاشته ام تا توضیح آن راحتتر باشد

.برای توضیح پیاده سازی به کامنت های داخل کد توجه شود.

```
    def runValueIteration(self):
        "*** YOUR CODE HERE ***"
        states = self.mdp.getStates()
        predecessors = util.Counter()

        for state in states:
            if not self.mdp.isTerminal(state):
                predecessors[state] = set()

        for state in states:    # step 1: find all the predecessors of the 'state'
            if not self.mdp.isTerminal(state):
                for action in self.mdp.getPossibleActions(state):
                    T = self.mdp.getTransitionStatesAndProbs(state, action)
                    for nextState, prob in T:
                        if not self.mdp.isTerminal(nextState) and prob != 0:
                            predecessors[nextState].add(state)

        priority_queue = util.PriorityQueue()  # step 2: create an empty priority queue to keep each state's priority

        for state in states:    # step 3: push states into the priority queue with the priority of -diff
            if not self.mdp.isTerminal(state):
                Q = self.computeQValueFromValues(state, self.getPolicy(state))
```

```
                Q = self.computeQValueFromValues(state, self.getPolicy(state))
                diff = abs(Q - self.values[state])
                priority_queue.push(state, -diff)

    for i in range(self.iterations):  # step 4: do iterations
        if priority_queue.isEmpty():
            return

        state = priority_queue.pop()  # note that the 'state' would not be a terminal state as we checked when we created the predecessors

        self.values[state] = self.computeQValueFromValues(state, self.getPolicy(state))

        for predecessor in predecessors[state]:  # update each predecessor's priority of the state
            Q = self.computeQValueFromValues(predecessor, self.getPolicy(predecessor))
            diff = abs(Q - self.values[predecessor])

            if diff > self.theta:
                priority_queue.update(predecessor, -diff)
```

6- یادگیری

در getQvalue, Q_value استیت داده شده با اکشن موردنظر برگردانده می شود. این اطلاعات در لیست self.qvalue قرار دارد

در computeActionFromQValue بهترین حرکت را با استفاده از لیستی را از QValue ها و اکشن متناظر با آن در این استیت بدست می آورد
در getAction یکی از بهترین حرکات بر گردانده می شود
در computeValueFromQValues از روی value_Q های یک استیت می توان value آن را محسابه کرد و روش دیگر برای بدست آوردن value این است که از روی سیاست آن استیت بتوانیم value را بدست آوریم(باتوجه به فرمول اسلایدها)
وقتی هیچ اکشن مجازی وجود ندارد باید مقدار 0.0 را برگرداند
و در تابعupdate ، Q_valueرا با استفاده از فرمول آپدیت می کنیم.

```
"""
def __init__(self, **args):
    "You can initialize Q-values here..."
    ReinforcementAgent.__init__(self, **args)

    "*** YOUR CODE HERE ***"
    self.values = util.Counter()

def getQValue(self, state, action):
    """
      Returns Q(state,action)
      Should return 0.0 if we have never seen a state
      or the Q node value otherwise
    """
    "*** YOUR CODE HERE ***"
    return self.values[state, action]
```

```python
    def computeValueFromQValues(self, state):
        """
          Returns max_action Q(state,action)
          where the max is over legal actions.  Note that if
          there are no legal actions, which is the case at the
          terminal state, you should return a value of 0.0.
        """
        "*** YOUR CODE HERE ***"
        legalActions = self.getLegalActions(state)
        if len(legalActions) == 0:
            return 0.0
        return self.getQValue(state, self.getPolicy(state))

    def computeActionFromQValues(self, state):
        """
          Compute the best action to take in a state.  Note that if there
          are no legal actions, which is the case at the terminal state,
          you should return None.
        """
        "*** YOUR CODE HERE ***"
        legalActions = self.getLegalActions(state)
        if len(legalActions) == 0:
            return None
        values = [self.getQValue(state, action) for action in legalActions]
        maxIndex = values.index(max(values))
        return legalActions[maxIndex]
```

.

```python
    def update(self, state, action, nextState, reward):
        """
          The parent class calls this to observe a
          state = action => nextState and reward transition.
          You should do your Q-Value update here

          NOTE: You should never call this function,
          it will be called on your behalf
        """
        "*** YOUR CODE HERE ***"
        S = reward + self.discount * self.getValue(nextState)
        Q = self.getQValue(state, action)
        newQ = Q + self.alpha * (S - Q)
        self.values[state, action] = newQ
```

```
*** PASS: test_cases\q6\1-tinygrid.test
*** PASS: test_cases\q6\2-tinygrid-noisy.test
*** PASS: test_cases\q6\3-bridge.test
*** PASS: test_cases\q6\4-discountgrid.test
Provisional grades
==================
Question q6: 4/4
------------------
Total: 4/4

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

توضیح دهید که اگر مقدار Q برای اقداماتی که عامل قبلا ندیده، بسیارکم یا بسیار زیاد باشد چه اتفاقی میافتد.

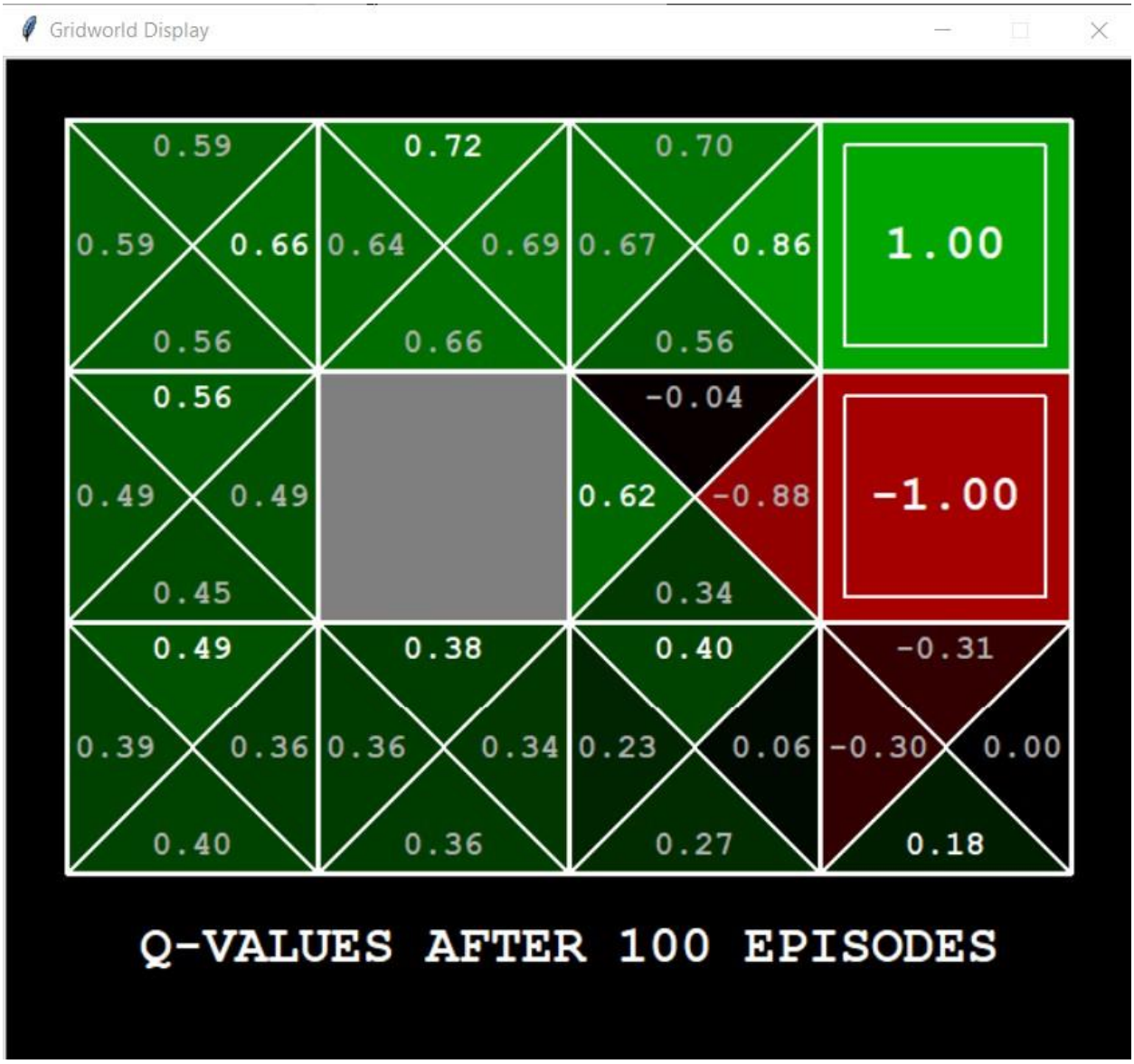
Q_value اولیه اگر بسیار کم باشد هیچ وقت اون اکشن انتخاب نمی شود تا عامل آن را تجربه کند بنابراین مقدار کمی باقی می ماند. و هنگامی که بسیار زیاد باشد بسیار طول می کشد تا عامل با اکشن هایش در جهت آن Q_value مقدار ش را کم کند.

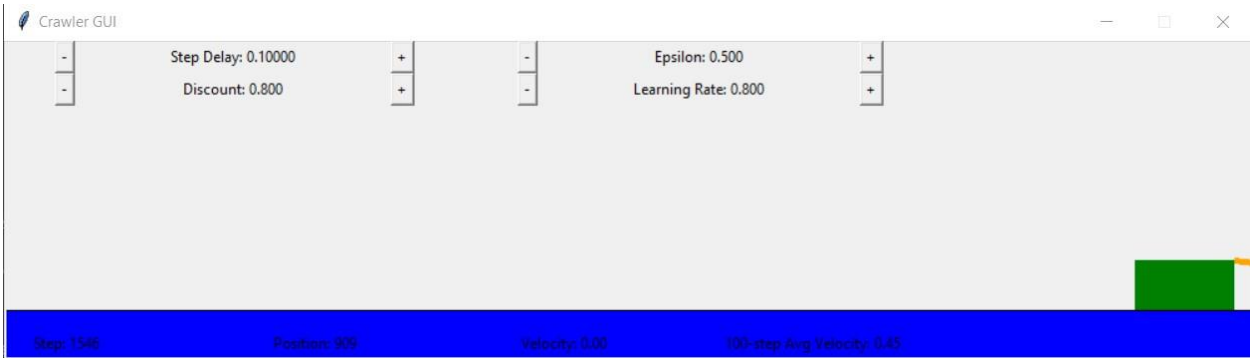
7- Epsilon حریصانه

ابتدا از تابع flipCoin استفاده کردم تا مشخص کنم انتخاب عامل به صورت تصادفی است یا خیر اگر تصادفی بود، با استفاده از choice.randomیک اکشن تصادفی از بین تمامی اکشن های مجاز انتخاب میشود .در غیر این صورت اکشنی انتخاب میشود که سیاست آنرا میگوید.

```
def getAction(self, state):
    """
      Compute the action to take in the current state.  With
      probability self.epsilon, we should take a random action and
      take the best policy action otherwise.  Note that if there are
      no legal actions, which is the case at the terminal state, you
      should choose None as the action.

      HINT: You might want to use util.flipCoin(prob)
      HINT: To pick randomly from a list, use random.choice(list)
    """
    # Pick Action
    legalActions = self.getLegalActions(state)
    action = None
    "*** YOUR CODE HERE ***"
    if len(legalActions) !=0:
        if util.flipCoin(self.epsilon):
         action = random.choice(legalActions)
        else:
         action = self.getPolicy(state)
    return action
```


Gridworld Display
0.59
0.59
0.66
0.56
0.72
0.64
0.69
0.66
0.70
0.67
0.86
0.56
1.00
0.56
0.49
0.49
0.45
-0.04
0.62
-0.88
0.34
-1.00
0.49
0.39
0.36
0.40
0.38
0.36
0.34
0.36
0.40
0.23
0.06
0.27
-0.31
-0.30
0.00
0.18
Q-VALUES AFTER 100 EPISODES


Crawler GUI
Step Delay: 0.10000
Discount: 0.800
Epsilon: 0.500
Learning Rate: 0.800

8- بررسی دوباره عبور از پل

ایده ای که در بخش اول استفاده کرده بودم این بود که نویز را کم کنم تا حرکت تصادفی عامل به کمترین مقدار خود برسد (noise=0.003) اما در اینجا با توجه به اینکه عامل در نیمی از حالت ها به صورت تصادفی انتخاب میکند عامل با آن ایده ای که در بخش از آن بهره بردم متناقض میشود چون انگاری که نویز زیادی در انتخاب اکشن عامل وجود دارد .پس مقدار Possible Notرا برگرداندم و جوابی برای آن پیدا نکردم.

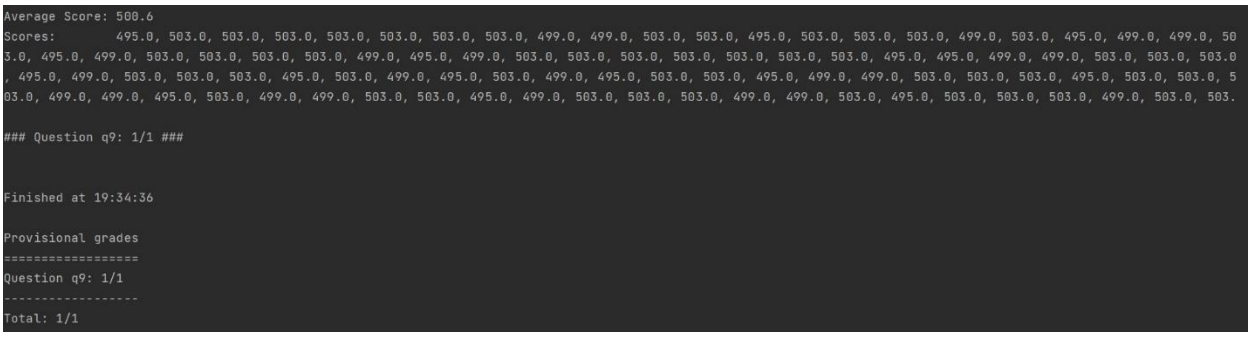
```
### Question q8: 1/1 ###


Finished at 19:30:05

Provisional grades
==================
Question q8: 1/1
------------------
Total: 1/1
```

9- پک من و Q_Learning

با توجه به پیاده سازی های قبلی این بخش بدون تغییر خاصی توانست نمره لازمه را بدست بیاورد.



10- یادگیری تقریبی

در این روش با استفاده از فرمول هایی که در دستورکار آورده شده است توانستم پیاده سازی لازمه را انجام دهم.

getQValue

خروجی این تابع مقدار جمع wi * featurei است که با توجه به راهنمایی های آورده شده در کد خروجی بدست آورده شده است .

Update

در این تابع ابتدا بردار ویژگی ها را بدست می آوریم و با توجه به مقدار قبلی Q و مقدار جدید آن، میزان پیشرفت در هر مرحله که diff نامیده می شود را بدست میاوریم

سپس به ازای هر ویژگی وزن مربوطه را با توجه به مقدار alpha و diff بروزرسانی می کنیم.

python pacman.py-p ApproximateQAgent-x 2000-n 2010-l smallGrid

```
Reinforcement Learning Status:
       Completed 1600 out of 2000 training episodes
       Average Rewards over all training: -100.18
       Average Rewards for last 100 episodes: 227.20
       Episode took 1.85 seconds
       Average Rewards over all training: -37.27
       Average Rewards for last 100 episodes: 207.14
       Episode took 1.42 seconds
Training Done (turning off epsilon and alpha)
-----------------------------------------
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Average Score: 501.0
Scores:        503.0, 495.0, 503.0, 499.0, 499.0, 503.0, 503.0, 499.0, 503.0, 503.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```

python pacman.py-p ApproximateQAgent-a extractor=SimpleExtractor-x 50-n 60-l mediumGrid

```
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 525
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 525
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 529
Pacman emerges victorious! Score: 527
Pacman emerges victorious! Score: 527
Average Score: 527.2
Scores:        527.0, 529.0, 525.0, 529.0, 527.0, 525.0, 527.0, 529.0, 527.0, 527.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```

```
### Question q10: 3/3 ###


Finished at 20:20:15

Provisional grades
==================
Question q10: 3/3
------------------
Total: 3/3
```