دانشگاه صنعتی اصفها ن
دانشکده برق و کامپیوتر

گزارش درس یادگیری تقویتی- تکلیف اول

دکتر مازیار پالهنگ

دانشجو :

پردیس مرادبیکی

شماره دانشجویی :

٤٠٠٠١٥٦٥

نیمسال دوم ١٤٠١-١٤٠٠

فهرست مطالب

فهرست شکل‌ها

گزارش تکلیف آشنائی با برنامه نویسی پویا در یادگیری تقویتی

۱- تمرین اول :

متن سوال یک: برنامه‌ای به زبان پایتون بنویسید و محیط بازی را طراحی کنید و با استفاده از الگوریتم ارزیابی سیاست، سیاستی که انجام هر یک از حرکات احتمال یکسان دارد را ارزیابی نموده و مقادیر آن را نمایش دهید. ضریب تخفیف را ۰.۹ در نظر بگیرید.

شرح برنامه تمرین اول:

```
#تمرین شماره 1: استفاده از الگوریتم ارزیابی سیاست جهت یافتن مقادیر حالات


#کتابخانه مورد نیاز
import numpy as np
```

شکل۱- فراخوانی کتابخانه

شرح خطوط کد:

کتابخانه‌ای که جهت اجرای این برنامه فراخوانی می‌شود، به شرح زیر است:

- کتابخانه numpy برای انجام اعمال ریاضی بر روی آرایه ها و ماتریس ها

```
: #ساخت محیط
P=dict()
for s in S: # برای کلیه حالات
    P[s]=dict()
    # اگر عامل در خانه هدف قرار گیرد به ازای هر عمل حالت بعدی را همان خانه و احتمال رسیدن
    #را 0.25 و پاداش عامل در همان حالت را 100 در نظر بگیرد
    if (terminal_state(s)):
        P[s][up]=(s,0.25,100)
        P[s][down]=(s,0.25,100)
        P[s][right]=(s,0.25,100)
        P[s][left]=(s,0.25,100)
    else:# یعنی عامل به هدف نرسیده

    #اگر عامل در 8 خانه انتها به هدف است یعنی در ضلع بالای مربع و عمل بالا رفتن میخواهد انجام دهد
    # حالت بعدی را همان حالت در نظر بگیرد و از مربع 8 در 8 بیرون نرود
        next_s= s if(64-s<=8) else s+8

        #اگر عامل با عمل بالا رفتن به یکی از حالت های دیوار برخورد میکند در همان خانه باقی بماند
        if next_s in wall:
            P[s][up]=(s,0,0)

        #در غیر این صورت 8 خانه به خانه ای که در ان قرار دارد اضافه شود
        else:
            P[s][up]=(next_s,0.25,reward)

    #اگر عامل در 8 خانه ابتدایی است یعنی در ضلع پایین مربع و عمل پایین رفتن میخواهد انجام دهد
    # حالت بعدی را همان حالت در نظر بگیرد و از مربع 8 در 8 بیرون نرود
        next_s= s if(s<8) else s-8

        #اگر عامل با عمل پایین رفتن به یکی از حالت های دیوار برخورد میکند در همان خانه باقی بماند
        if next_s in wall:
            P[s][down]=(s,0,0)

        #در غیر این صورت 8 خانه از خانه ای که در ان قرار دارد کم شود
        else:
            P[s][down]=(next_s,0.25,reward)

    #اگر عامل در خانه های سمت چپی است یعنی در ضلع سمت چپی مربع و عمل چپ رفتن میخواهد انجام دهد
    # حالت بعدی را همان حالت در نظر بگیرد و از مربع 8 در 8 بیرون نرود
        next_s= s if((s+1)%8==0) else s+1

        #اگر عامل با عمل چپ رفتن به یکی از حالت های دیوار برخورد میکند در همان خانه باقی بماند
        if next_s in wall:
            P[s][left]=(s,0,0)

        #در غیر این صورت 1 خانه از خانه ای که در ان قرار دارد کم شود
        else:
            P[s][left]=(next_s,0.25,reward)

    #اگر عامل در خانه های سمت راستی است یعنی در ضلع سمت راست مربع و عمل راست رفتن میخواهد انجام دهد
    # حالت بعدی را همان حالت در نظر بگیرد و از مربع 8 در 8 بیرون نرود
        next_s= s if(s%8==0) else s-1

        #اگر عامل با عمل راست رفتن به یکی از حالت های دیوار برخورد میکند در همان خانه باقی بماند
        if next_s in wall:
            P[s][right]=(s,0,0)

        #در غیر این صورت 1 خانه به خانه ای که در ان قرار دارد اضافه شود
        else:
            P[s][right]=(next_s,0.25,reward)

        #در خانه هایی که دیوار در ان قرار دارد صفر در نظر بگیر#
for s in S:
    if s in wall:
        P[s][left]=(s,0,0)
        P[s][right]=(s,0,0)
        P[s][down]=(s,0,0)
        P[s][up]=(s,0,0)
```

شکل۲- ایجاد محیط شبکه ای

شرح خطوط کد:

در یادگیری تقویتی[1]، محیط در واقع محلی است که عامل در آن زندگی و تعامل می‌کند.

محیط باتوجه به صورت سئوال، شبکه‌ای ۸*۸ است که حاصل آن ۶۴ موقعیت[2]یا حالت ممکن برای عامل است. این ۶۴ حالت بخشی از فضای حالت[3] محسوب می‌شوند. عامل[4] تنها موجود در این شبکه است که در حال حرکت به سمت نقطه پایانی[5] است. ابتدا عامل از موقعیت ۲۴ شروع به حرکت میکند.

[1] Reinforcement learning
[2] state

عمل معمولا بر پایه محیط است و محیط‌های متفاوت منجر به اعمال متفاوتی می‌شوند. اعمال معمولا تعداد مشخصی دارند. در اینجا ۴ عمل بالا، پایین، چپ و راست را می‌توان در هر حالت انجام داد که به ترتیب اعداد ۰-۳ را به آنها نسبت داده شده است.

در محیط شبکه‌ای مطرح شده در صورت سئوال، موقعیت‌هایی تحت عنوان دیوار مطرح شده که عامل حق ورود به آن‌ها را ندارد و در صورتی که حرکتی برای رفتن به آن موقعیت ها داشته باشد، در همان موقعیت باید بماند.

در این قسمت برای تمام ۶۴ موقعیت که زمین ۸*۸ دارد اگر هدف باشد پاداش خروجی را ۱۰۰ در نظر میگیریم و بقیه پاداش صفر دارند و با هر حرکت عامل در محیط موقعیت عامل با توجه به عمل انجام شده در محیط تغییر پیدا می‌کند.

```
P
{0: {0: (8, 0.25, 0), 1: (0, 0.25, 0), 2: (1, 0.25, 0), 3: (0, 0.25, 0)},
 1: {0: (9, 0.25, 0), 1: (1, 0.25, 0), 2: (2, 0.25, 0), 3: (0, 0.25, 0)},
 2: {0: (10, 0.25, 0), 1: (2, 0.25, 0), 2: (3, 0.25, 0), 3: (1, 0.25, 0)},
 3: {0: (11, 0.25, 0), 1: (3, 0.25, 0), 2: (4, 0.25, 0), 3: (2, 0.25, 0)},
 4: {0: (4, 0, 0), 1: (4, 0.25, 0), 2: (5, 0.25, 0), 3: (3, 0.25, 0)},
 5: {0: (13, 0.25, 0), 1: (5, 0.25, 0), 2: (6, 0.25, 0), 3: (4, 0.25, 0)},
 6: {0: (14, 0.25, 0), 1: (6, 0.25, 0), 2: (7, 0.25, 0), 3: (5, 0.25, 0)},
 7: {0: (15, 0.25, 0), 1: (7, 0.25, 0), 2: (7, 0.25, 0), 3: (6, 0.25, 0)},
 8: {0: (16, 0.25, 0), 1: (0, 0.25, 0), 2: (9, 0.25, 0), 3: (8, 0.25, 0)},
 9: {0: (17, 0.25, 0), 1: (1, 0.25, 0), 2: (10, 0.25, 0), 3: (8, 0.25, 0)},
 10: {0: (18, 0.25, 0), 1: (2, 0.25, 0), 2: (11, 0.25, 0), 3: (9, 0.25, 0)},
 11: {0: (19, 0.25, 0), 1: (3, 0.25, 0), 2: (11, 0, 0), 3: (10, 0.25, 0)},
 12: {0: (12, 0, 0), 1: (12, 0, 0), 2: (12, 0, 0), 3: (12, 0, 0)},
 13: {0: (21, 0.25, 0), 1: (5, 0.25, 0), 2: (14, 0.25, 0), 3: (13, 0, 0)},
 14: {0: (22, 0.25, 0), 1: (6, 0.25, 0), 2: (15, 0.25, 0), 3: (13, 0.25, 0)},
 15: {0: (23, 0.25, 0), 1: (7, 0.25, 0), 2: (15, 0.25, 0), 3: (14, 0.25, 0)},
 16: {0: (24, 0.25, 0), 1: (8, 0.25, 0), 2: (17, 0.25, 0), 3: (16, 0.25, 0)},
 17: {0: (25, 0.25, 0), 1: (9, 0.25, 0), 2: (18, 0.25, 0), 3: (16, 0.25, 0)},
 18: {0: (18, 0, 0), 1: (10, 0.25, 0), 2: (19, 0.25, 0), 3: (17, 0.25, 0)},
```

شکل۳- خروجی محیط شبکه ای ایجاد شده : P[state][action]

شرح شکل۳:

همانطور که در شکل مشاهده می‌کنید،عامل اگر در حالت ۰ در محیط باشد با انتخاب عمل بالا رفتن موقعیت بعدی عامل در محیط موقعیت ۸ خواهد بود و با احتمال ۰.۲۵ درصد ممکن است عامل عمل بالا رفتن را انتخاب کند و پاداش حرکت عامل در محیط چون به حالت پایانی نمیرسد برابر صفر قرار می‌گیرد. همچنین با انتخاب حرکت به سمت پایین عامل در همان خانه ۰ باقی خواهد ماند.

[3] Set of states
[4] agent
[5] goal

```python
#(محاسبه تابع مقدار(ارزیابی سیاست
def policy_evaluation(P,policy,threshold,dicount):
    value=np.zeros((noS,)) # مقداردهی اولیه تابع مقدار
    while True: # تا زمانی که همگرایی در تابع مقدار اتفاق نمی افتد حلقه را ادامه بده
        new_value=np.zeros((noS,)) # v مقداردهی اولیه تابع بروزرسانی

        change=0
        for s in S: # برای تمام حالات
            v=0

            for a,action_prob in enumerate(policy[s]): #  برای تمامی اعمال و احتمال اعمال تحت سیاست مشخص در آن حالت
                next_state,probability,reward=P[s][a] # برای 4 مقداری که از بودن عامل در آن حالت و انجام آن عمل در محیط
                temp=probability*action_prob*(reward+discount*value[next_state]) # بروزرسانی تابع مقدار
                v+=temp

            change=max(change,np.abs(v-value[s])) # محاسبه اختلاف تابع مقدار جدید و قدیم
            new_value[s]=v # در صورتی که شرط خاتمه برقرار، قرار دادن تابع مقدار در مقدار جدید

        if change < threshold:
              break

        value=new_value

    return value
```

شکل۴- ایجاد تابع جهت محاسبه تابع مقدار حالت یا ارزیابی سیاست

شرح خطوط کد:

برای تمامی موقعیت‌ها، مقدار بروزرسانی تابع ارزش را در آرایه value قرار می‌دهیم . بعبارت دیگر بروزرسانی ها به ازاء تمام توابع ارزشی که در هر موقعیت وجود دارد صورت می‌پذیرد وتا جایی ادامه می‌یابد که تابع ارزش به یک مقدار واحد همگرا شود (شرط خاتمه). بروزرسانی تابع طبق فرمول معادله ۱ صورت می گیرد.

$$V(s) \leftarrow \sum_{a\in\mathcal{A}(s)} \pi(a|s) \sum_{s'\in\mathcal{S},r\in\mathcal{R}} p(s',r|s,a)(r+\gamma V(s')) \qquad \text{(معادله ۱)}$$

همانگونه که در معادله ۱ دیده می‌شود، برای تشکیل دو سیگما، دو حلقه نیاز است. در حلقه اول، مقدار احتمال عمل و عمل انتخابی تحت سیاستpolicy random را در موقعیت s نیاز داریم، که این حلقه برای تمام اعمال موجود در آن موقعیت محاسبه می‌شود. در حلقه دوم، از تابع enurmerate، استفاده شده است که علاوه بر مقدار، اندیس آن را نیز برمی‌گرداند. به بیانی دیگر، به هر عملی یک اندیس نسبت داده می‌شود. در این حلقه رفتن به موقعیتی تحت عمل action، ۳ مقدار که شامل: موقعیت بعدی، احتمال انتقال از موقعیتی به موقعیت دیگر و پاداش است را بازمی‌گرداند.

در نهایت با محاسبه بزرگترین اختلاف بین مقادیر تابع ارزش فعلی با قبلی با آستانه تعریف شده، بررسی می‌شود که آیا تابع مقدار حالت به عدد واحدی در هر حالت همگرا شده است یا خیر(بررسی شرط توقف).

```python
random_policy = np.ones([64, 4])/4 #  سیاست انتخاب عمل تصادفی

threshold = 0.0000000000001 # تعیین پارامتری با مقدار بسیار کم برای بررسی همگرایی
discount = 0.9  #مقداردهی اولیه ضریب تخفیف

random_policy_value=policy_evaluation(P,random_policy,threshold,discount)
```

شکل۵-تعیین پارامترها و فراخوانی تابع ارزیابی سیاست

شرح خطوط کد:

در الگوریتم ارزیابی سیاست[6]، سیاستی را بعنوان سیاست تصادفی با احتمال یکنواخت با توجه به فرمول معادله ۲ تعریف و مقداردهی اولیه می‌کنیم.

$$\pi(a|s) = \frac{1}{|A(s)|} \quad s \in \mathrm{S}, \mathrm{a} \in \mathrm{A(s)} \qquad \text{(معادله 2)}$$

ماتریس مربوط به سیاست یک ماتریس دو بعدی است. ماتریس ۴*۶۴ که تعداد سطرهای آن تعداد موقعیت‌های موجود در این محیط (۶۴) و ستون‌هایش تعداد اعمال (۴) است.

پارامترها در این برنامه به شرح زیر است:

- discount: ضریب تخفیف[7] است. عددی بین ۰ و ۱ که در این مسئله میزان تاثیر پاداش‌های فعلی را به ازاء رفتن از حالت s به حالت s´ با انتخاب عمل a، تقریبا همانند پاداش‌های قبلی در نظر می‌گیریم(۱=γ).
- threshold: پارامتری برای بررسی همگرایی است. برای اینکه بفهمیم جدول مقدار نسبت به تکرار قبلی تغییر می‌کند یا خیر، میتوانیم تفاوت بین جدول مقدارهای به دست آمده از تکرار قبلی و جدول مقدار به دست آمده از تکرار فعلی را محاسبه کرد. اگر تفاوت بسیار کوچک باشد یعنی تفاوت کمتر از یک آستانه بسیار کوچک باشد آنگاه میتوانیم بگوییم که به همگرایی رسیدیم زیرا تغییر زیادی در تابع مقدار حالت وجود ندارد.

۱-۱ خروجی کد تمرین اول:

```
Value Function for policy: all actions equiprobable: discount = 1
[[1.10835234e-02 2.57063013e-03 3.62480629e-04 4.09475248e-05
  2.74419062e-06 2.15334473e-07 8.00480756e-08 2.29596501e-07]
 [1.52598697e-01 2.71134480e-02 2.82563178e-03 2.48988052e-04
  0.00000000e+00 4.05778433e-07 7.55790178e-07 3.13430296e-06]
 [2.25078348e+00 2.75820209e-01 1.74851919e-02 1.11722953e-03
  7.07721200e-05 5.52133030e-06 8.47251340e-06 4.60291577e-05]
 [3.33333333e+01 2.11774122e+00 0.00000000e+00 7.07204186e-05
  9.60306142e-06 8.69087296e-06 8.32539362e-05 6.78830549e-04]
 [2.25070452e+00 2.74705919e-01 0.00000000e+00 4.69410638e-06
  3.46557113e-06 4.06756395e-05 6.36069045e-04 1.00531751e-02]
 [1.52528603e-01 2.68489624e-02 0.00000000e+00 9.19712483e-07
  4.76330848e-07 2.58474271e-06 0.00000000e+00 1.49482728e-01]
 [1.03755661e-02 2.34887625e-03 1.48900242e-04 9.54496252e-06
  6.51267254e-07 2.03913050e-07 0.00000000e+00 2.23218774e+00]
 [7.56011243e-04 2.08591343e-04 2.39826509e-05 2.24817887e-06
  1.95069663e-07 2.65988469e-08 0.00000000e+00 3.33333333e+01]]
```

(الف)

[6] Policy evaluation
[7] Discount factor

```
Value Function for policy: all actions equiprobable: discount = 0.1:
[[6.42202017e-06 1.59305593e-07 2.23923492e-09 2.49032164e-11
  1.56287570e-13 1.04773790e-15 2.91038305e-16 1.00117177e-14]
 [1.01451988e-03 1.89053313e-05 1.96708545e-07 1.72047160e-09
  0.00000000e+00 1.97323970e-14 3.94647941e-14 1.58773037e-12]
 [1.61283334e-01 2.00997711e-03 1.25640773e-05 7.85423702e-08
  4.90947277e-10 3.12743941e-12 4.72279498e-12 2.52427533e-10]
 [2.56410256e+01 1.60281534e-01 0.00000000e+00 4.90946986e-10
  6.17624028e-12 4.74153785e-12 5.00112714e-10 4.01297739e-08]
 [1.61283334e-01 2.00989857e-03 0.00000000e+00 3.07871960e-12
  1.61584467e-12 2.49292410e-10 3.98789437e-08 6.37988184e-06]
 [1.01451962e-03 1.89036015e-05 0.00000000e+00 5.78584149e-14
  2.03726813e-14 1.55816087e-12 0.00000000e+00 1.01432120e-03]
 [6.38187461e-06 1.58048196e-07 9.87926265e-10 6.17559999e-12
  3.87080945e-14 9.95351002e-15 0.00000000e+00 1.61270692e-01]
 [4.03995072e-08 1.24818448e-09 1.40633783e-11 1.27009116e-13
  9.89530236e-16 5.82076609e-17 0.00000000e+00 2.56410256e+01]]
```

(ب)

```
Value Function for policy: all actions equiprobable: discount = 0.9 :
[[7.53796270e-03 1.58482553e-03 2.00920706e-04 2.03786808e-05
  1.21969276e-06 8.50533217e-08 3.07937181e-08 9.87921191e-08]
 [1.17347475e-01 1.88509671e-02 1.76579875e-03 1.39768579e-04
  0.00000000e+00 1.76519263e-07 3.32804725e-07 1.52792639e-06]
 [1.94244093e+00 2.14430205e-01 1.22003213e-02 6.98597305e-04
  3.97192704e-05 2.72026219e-06 4.18128908e-06 2.52036126e-05]
 [3.22580645e+01 1.83860032e+00 0.00000000e+00 3.96984919e-05
  4.80279478e-06 4.28313804e-06 4.60773486e-05 4.17151396e-04]
 [1.94239717e+00 2.13733180e-01 0.00000000e+00 2.35086624e-06
  1.68211798e-06 2.25442707e-05 3.93537042e-04 6.92759246e-03]
 [1.17310360e-01 1.87034913e-02 0.00000000e+00 4.12567750e-07
  2.06387854e-07 1.28473754e-06 0.00000000e+00 1.15418918e-01]
 [7.10648603e-03 1.46297211e-03 8.32294667e-05 4.77728369e-06
  2.89694153e-07 8.91198198e-08 0.00000000e+00 1.92954537e+00]
 [4.57711624e-04 1.15186250e-04 1.18855714e-05 9.97759192e-07
  7.73269274e-08 9.92066684e-09 0.00000000e+00 3.22580645e+01]]
```

(ج)

شکل۶- مقادیر خروجی الگوریتم ارزیابی سیاست در تمرین اول به ازاء مقادیر مختلف

ضریب تخفیف: الف) ضریب تخفیف=۱، ب) ضریب تخفیف=۰.۱، ج) ضریب تخفیف=۰.۹

نتیجه خروجی کد تمرین اول: همانطور که در شکل ۶ مشاهده می‌شود با کاهش مقادیر ضریب تخفیف تعداد خانه‌هایی که دارای مقدار غیر صفر می‌شوند،کاهش می‌یابد. دلیل این امر آن است که به پاداش‌های فوری اهمیت بیشتری داده می‌شود. بنابراین، مقادیر ارزش خانه‌های نزدیک به خانه‌ی هدف بروزرسانی می‌شوند. با افزایش مقدار ضریب تخفیف تعداد خانه‌های بیشتری بروزرسانی می‌شود(مکان این خانه‌ها با توجه به یادگیری که در عامل اتفاق افتاده است، تحت سیاست داده شده قرار دارد). همچنین همانطور که در شکل مشاهده می‌کنید حالت‌هایی که در بازی دیوار بوده اند در اینجا حفره شناسایی شده و مقادیر در آن صفر در نظر گرفته شده است.

۲- تمرین دوم:

متن سوال دوم: برنامه ای به زبان پایتون بنویسید و با استفاده از الگوریتم تکرار سیاست با ضریب تخفیف ۰.۹ سیاست بهینه و مقادیر تابع مقدار حالت بهینه را یافته و نمایش دهید. آیا برای یافتن سیاست بهینه لازم است که تابع مقدار حالت بهینه را بیابید؟ (سعی کنید سیاست را بصورت گرافیکی نمایش دهید، بطور مثال با علائم V، >،< و ^.)

شرح برنامه تمرین دوم:

```
#تمرین شماره 2: استفاده از الگوریتم تکرار سیاست جهت یافتن سیاست و تابع مقدار بهینه

#کتابخانه مورد نیاز
import numpy as np
```

شکل۷- فراخوانی کتابخانه

شرح خطوط کد:

کتابخانه‌ی مورد نیاز جهت اجرای این برنامه که فراخوانی می‌شود، به شرح زیر است:

- کتابخانه numpy برای انجام اعمال ریاضی بر روی آرایه ها و ماتریس ها

```
#(محاسبه تابع مقدار(ارزیابی سیاست
def policy_evaluation(P,policy,threshold,dicount):
    value=np.zeros((noS,)) # مقداردهی اولیه تابع مقدار
    while True: # تا زمانی که همگرایی در تابع مقدار اتفاق نمی افتد حلقه را ادامه بده
        new_value=np.zeros((noS,)) # v مقداردهی اولیه تابع بروزرسانی

        change=0
        for s in S: # برای تمام حالات
            v=0

            for a,action_prob in enumerate(policy[s]): #   برای تمامی اعمال و احتمال اعمال تحت سیاست مشخص در آن حالت
                next_state,probability,reward=P[s][a] # برای 4 مقداری که از بودن عامل در آن حالت و انجام آن عمل در محیط
                temp=probability*action_prob*(reward+discount*value[next_state]) # بروزرسانی تابع مقدار
                v+=temp

            change=max(change,np.abs(v-value[s])) # محاسبه اختلاف تابع مقدار جدید و قدیم
            new_value[s]=v # در صورتی که شرط خاتمه برقرار، قرار دادن تابع مقدار در مقدار جدید

        if change < threshold:
            break

        value=new_value

    return value
```

شکل۸- ایجاد تابع policy_evaluation جهت محاسبه تابع مقدار

شرح خطوط کد:

در شکل۷ تابع مقدار حالت با ورودی سیاست محاسبه می‌شود که به شرح زیر است(الگوریتم ارزیابی سیاست):

به تابع مقداری که مقادیر حالت را مشخص می‌کند(value) بصورت اولیه مقداردهی می‌کنیم. با توجه به اینکه در این مسئله با ۶۴ حالت مختلف مواجه هستیم، بنابراین آرایه‌ای با ۶۴ عنصر می‌باشد که با مقدارصفر مقداردهی اولیه شده است.

پارامتر آستانه: دارای مقدار بسیار کوچکی است و جهت تعیین همگرایی از آن استفاده می‌کنیم. توضیح آنکه بدلیل محاسبه مقدار تابع در هر حالت، باید اختلاف دو تابع مقدار حالت قبلی (دارای ۶۴ مقداری) با مقدار حالت فعلی (آرایه‌ای ۶۴ مقداری) محاسبه شود و در صورتی که جمع این اختلافات از آستانه کمتر باشد، یعنی تابع مقدار حالت همگرا به یک عدد شده است و دیگر به محاسبه این حلقه نیازی نیست.

- ایجاد حلقه‌ای که تا همگرایی ادامه دارد.
- ایجاد آرایه‌ای جهت قرار دادن مقادیرتوابع مقدار بروزرسانی شده.
- با عبور از هر حالت، عمل (که سیاست مشخص در هر حالت آن را تعیین می‌کند) تعیین می‌شود و سپس تابع مقدار حالت برای هر حالت محاسبه می‌شود. در نهایت با استفاده از تابع بهینگی بلمن معادله ۳، که در آن مقادیر فعلی به مقادیر قبلی تابع حالت وابسته است، برای تمامی فضای حالت تابع مقدار حالت را بروزرسانی می‌کنیم. مقادیر حالت بعد، ماتریس انتقال[8]و پاداش به ازاء انتقال‌ها با توجه به دستور موجود در این محیط تحت اعمال مختلف در تمامی حالات محاسبه می‌شود.

$$V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right] \qquad \text{(معادله ۳)}$$

توضیح تکمیلی۱:

عامل با انجام اقداماتی برای تغییر یک حالت به دیگری، واکنش نشان می‌دهد. پس از تغییر در حالت، بسته به اقدامی که انجام می‌شود به عامل پاداش[9] یا مجازات[10] داده می‌شود . تابع پاداش نقش اساسی در تنظیم، بهینه‌سازی الگوریتم و متوقف کردن الگوریتم آموزش دارد. این امر بستگی به وضعیت کنونی، اقدامی که انجام شده و حالت بعدی محیط دارد.

توضیح تکمیلی۲:

مقدار ضریب تخفیف عددی بین ۰ و ۱ است، در این مسئله ۰.۹ در نظر گرفته شده است. این بدان معنی است که تاثیر پاداش‌های فعلی همانند پاداش‌های قبلی است.

[8] Transition matrix
[9] Reward
[10] Penalty

```
#الگوریتم تکرار سیاست
def policy_iteration(P,discount,threshold):

    value=np.zeros((noS,)) #مقدار دهی اولیه تابع مقدار
    policy=np.ones([64, 4])/4 # مقدار دهی اولیه سیاست

    while True:
        #محاسبه تابع مقدار با ارزیابی سیاست
        value=policy_evaluation(P,policy,threshold,discount) #فراخوانی تابع مقدار حالت

        new_value=np.zeros((noS,)) #مقدار دهی اولیه
        new_policy=np.zeros([noS,4]) #مقدار دهی اولیه

        #محاسبه استخراج سیاست با بهبود سیاست
        policy_stable=True
        for s in S: #محاسبه Q برای هر حالت

            old_action=policy[s]
            action_values = np.zeros(noA)
            for a in range(noA):   #محاسبه Q برای هر عمل
                    next_state,probability,reward = P[s][a] # بدست آوردن دانشMDP
                    action_values[a] += probability*(reward + discount*value[next_state]) # Q محاسبه و بروز رسانی مقادیر
            max_total = np.amax(action_values)   #  انتخاب بیشترین مقدار تابع عمل وحالت - انتخاب سیاست حریصانه
            best_a = np.argmax(action_values)

            new_policy[s][best_a]=1 #قرار دادن مقدار یک مقابل عملی که بهینه است

            new_value[s]=max_total
            if (np.array_equal(old_action,new_policy[s])!=True):
                policy_stable=False

        value=new_value

        if policy_stable: #مقدار برای مقادیری که دیوار است
            value[wall]=12,26,34,42,46,54,62
            return new_policy,value  #  برگرداندن مقدار سیاست بهینه
        else:
            policy=new_policy # جایگزین کردن سیاست قبلی با سیاست بعدی
```

شکل۹- ایجاد تابع policy_iteration جهت محاسبه سیاست بهینه

شرح خطوط کد:

در اینجا پس از فراخوانی تابع محاسبه تابع مقدار با مقدار دهی اولیه‌ی سیاست ، محاسبه سیاست، با ورودی مقادیر حالت(الگوریتم بهبود سیاست) را داریم :

- ابتدا مقادیر سیاست را با صفر مقدار دهی اولیه می‌کنیم. در این الگوریتم با بدست آوردن مقدار تابع مقدار-حالت عمل، سیاست را بصورت حریصانه بهبود می‌بخشیم.
- آرایه Q(تابع حالت-عمل) را بصورت اولیه مقداردهی می‌کنیم. این آرایه برای هر حالت محاسبه و مقادیر آن را برای هر عمل مجاز در آن حالت را تعیین کنیم.
- سپس بروزرسانی آرایه تابع حالت-عمل طبق معادله ۴ انجام می‌شود:

$$Q(s,a) \leftarrow \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s')) \qquad \text{(معادله ۴)}$$

بروزرسانی مقادیرQ در هر حالت، برطبق اعمال تعریف شده در این محیط (۴ عمل که بصورت ۰-۳ نمایش داده شده-اند) مشخص می‌شوند. در نهایت بزرگترین تابع حالت-عمل بعنوان سیاست در آن حالت، تعیین می‌شود. (سیاست بصورت حریصانه از بین توابع زوج‌های حالت-عمل تعیین می‌شود).

```
best_policy,corr_value=policy_iteration(P,discount,threshold) # فراخوانی تابع تکرار سیاست
```

شکل ۱۰- فراخوانی تابع policy_iteration جهت نمایش مقدار بهینه و سیاست بهینه

شرح خطوط کد:

- در الگوریتم تکرار سیاست، مقدار سیاست بهینه برای محیط شبکه‌ای ۸*۸ با ۴ عمل چپ،پایین، راست را چاپ می‌کنیم.
- همچنین برای سیاست بهینه‌ای که با الگوریتم تکرار سیاست بدست آمده است، با استفاده ازالگوریتم ارزیابی سیاست، تابع مقدار حالت آن را نیز محاسبه وچاپ می‌کنیم.

۲-۱ خروجی کد تمرین دوم:

همانگونه که در شکل ۱۱- الف دیده می‌شود مقادیر حالت با نزدیک شدن به خانه‌ی هدف بیشترین مقدار را به خود می‌گیرند. عملی که در سیاست بدست آمده در این الگوریتم، در شکل ۱۱-ب مشخص شده است، برطبق بیشترین مقدار عمل بدست آمده در تابع مقدار-حالت است(argmax روی تمامی اعمال).

```
corr_value  # چاپ تابع مقدار بهینه
```

```
array([[5.20833333e-01, 1.30208333e-01, 3.25520833e-02, 8.13802083e-03,
        2.03450521e-03, 5.08626302e-04, 5.08626302e-04, 2.03450521e-03],
       [2.08333333e+00, 5.20833333e-01, 1.30208333e-01, 3.25520833e-02,
        0.00000000e+00, 2.03450521e-03, 2.03450521e-03, 8.13802083e-03],
       [8.33333333e+00, 2.08333333e+00, 5.20833333e-01, 1.30208333e-01,
        3.25520833e-02, 8.13802083e-03, 8.13802083e-03, 3.25520833e-02],
       [3.33333333e+01, 8.33333333e+00, 0.00000000e+00, 3.25520833e-02,
        8.13802083e-03, 8.13802083e-03, 3.25520833e-02, 1.30208333e-01],
       [8.33333333e+00, 2.08333333e+00, 0.00000000e+00, 8.13802083e-03,
        8.13802083e-03, 3.25520833e-02, 1.30208333e-01, 5.20833333e-01],
       [2.08333333e+00, 5.20833333e-01, 0.00000000e+00, 2.03450521e-03,
        2.03450521e-03, 8.13802083e-03, 0.00000000e+00, 2.08333333e+00],
       [5.20833333e-01, 1.30208333e-01, 3.25520833e-02, 8.13802083e-03,
        2.03450521e-03, 2.03450521e-03, 0.00000000e+00, 8.33333333e+00],
       [1.30208333e-01, 3.25520833e-02, 8.13802083e-03, 2.03450521e-03,
        5.08626302e-04, 5.08626302e-04, 0.00000000e+00, 3.33333333e+01]])
```

(الف)

```
optimal_policy=
 [['^' '^' '^' '^' '>' '^' '^' '^']
 ['^' '^' '^' '^' '^' '^' '^' '^']
 ['^' '^' '>' '>' '>' '>' '^' '^']
 ['^' '>' '^' 'v' 'v' '^' '^' '^']
 ['v' 'v' '^' 'v' '<' '<' '<' '^']
 ['v' 'v' '^' '^' 'v' 'v' '^' '^']
 ['v' 'v' '>' '>' '>' 'v' '^' '^']
 ['v' 'v' 'v' 'v' 'v' 'v' '^' '^']]
```

(ب)

شکل ۱۱- خروجی کد تمرین دوم، الف) نمایش مقادیر بهینه و ب) نمایش مقادیرسیاست بهینه

نتیجه خروجی کد تمرین دوم: برای یافتن سیاست بهینه لازم نیست حتما تابع مقدار حالت بهینه را به دست آوریم زیرا میتوانیم با در نظر گرفتن آستانه کوچکتر به حالت بهینه دست یابیم.

۳- تمرین سوم

متن سوال سوم: با استفاده از الگوریتم تکرار مقدار با ضریب تخفیف ۰.۹، سیاست بهینه را بیابید. آیا از لحاظ اجرا تفاوتی بین این الگوریتم و الگوریتم تمرین شماره ۲ مشاهده کرده اید؟

شرح برنامه تمرین سوم

```
#تمرین شماره 3: استفاده از الگوریتم تکرار سیاست برای یافتن سیاست بهینه و مقدار بهینه


#کتابخانه مورد نیاز
import numpy as np
```

شکل۱۲- فراخوانی کتابخانه

شرح خطوط کد:

کتابخانه‌ی مورد نیاز جهت اجرای این برنامه که فراخوانی می‌شود، به شرح زیر است:

- کتابخانه numpy برای انجام اعمال ریاضی بر روی آرایه ها و ماتریس ها

```python
#الگوریتم تکرار مقدار

def value_iteration(P,discount,threshold):
    value=np.zeros((noS,)) #مقداردهی اولیه تابع مقدار
    while True:
        new_policy=np.zeros([noS,4]) # مقدار دهی اولیه سیاست
        change=0
        for s in S: # محاسبه برای تمامی حالات
            v=value[s]
            action_values = np.zeros(noA)
            for a in range(4):   #محاسبه هر عمل
                next_state,probability,reward = P[s][a] # MDPبدست آوردن دانش
                action_values[a] += probability*(reward + discount*value[next_state]) #محاسبه و بروز رسانی مقادیر
            max_total = np.amax(action_values)   # انتخاب بیشترین مقدار پاداش ---انتخاب حریصانه
            best_a = np.argmax(action_values)

            value[s]=max_total
            new_policy[s][best_a]=1 #قرار دادن مقدار یک مقابل عملی که بهینه است

            change=max(change,np.abs(v-value[s])) #محاسبه اختلاف تابع مقدار جدید و قدیم

        if change < threshold:
            break
    return new_policy,value.reshape(8,8)
```

شکل ۱۳- ایجاد تابع Value_iteration جهت محاسبه سیاست بهینه

شرح خطوط کد:

الگوریتم تکرار مقدار شامل دو مرحله است. در مرحله‌ی اول تابع مقدار بهینه را با انتخاب بیشترین پاداش بر روی تابع Q به دست می‌آوریم و محاسبه می‌کنیم و سپس در مرحله‌ی دوم سیاست بهینه را از تابع مقدار بهینه محاسبه و استخراج می‌کنیم.

در این جا ابتدا با مقدار دهی اولیه برابر صفر برای تابع مقدار شروع می‌کنیم. همچنین با قرار دادن مقدار اولیه برای سیاست، به ازای تمام حالت‌ها و اعمال تا همگرا شدن مقدار تابع بهینه، به وسیله مقادیری که از دانش MDP به دست آمده مقدار تابع را با معادله ۵ به دست می‌آوریم.

$$Q(s,a)= \sum_{s'} p(s'|s,a)\{R(s,a,s') + \gamma V(s')\} \qquad \text{معادله ۵}$$

بعد از به دست آوردن مقدار تابع به ازای هر عمل طبق سیاست حریصانه بیشترین مقدار را برمیگردانیم و آن را به عنوان عمل بهینه مشخص می‌کنیم.

```
best_policy,corr_value=value_iteration(P,discount,threshold)
```

شکل ۱۴ - فراخونی تابع value-iteration جهت نمایش مقدار بهینه و سیاست بهینه

شرح خطوط کد:

- در الگوریتم تکرار مقدار، مقدار سیاست بهینه برای محیط شبکه‌ای ۸*۸ با ۴ عمل چپ،پایین، راست را چاپ می‌کنیم.
- همچنین برای سیاست بهینه‌ای که با الگوریتم تکرار مقدار بدست آمده است، با استفاده ازالگوریتم ارزیابی سیاست، تابع مقدار حالت آن را نیز محاسبه وچاپ می‌کنیم.

۳-۱ خروجی کد تمرین سوم:

همانگونه که در شکل ۱۶ دیده می‌شود، مقادیر حالت با نزدیک شدن به خانه‌ی هدف بیشترین مقدار را به خود می-گیرند.

```
corr_value
```

```
array([[5.20833333e-01, 1.30208333e-01, 3.25520833e-02, 8.13802083e-03,
        2.03450521e-03, 5.08626302e-04, 5.08626302e-04, 2.03450521e-03],
       [2.08333333e+00, 5.20833333e-01, 1.30208333e-01, 3.25520833e-02,
        0.00000000e+00, 2.03450521e-03, 2.03450521e-03, 8.13802083e-03],
       [8.33333333e+00, 2.08333333e+00, 5.20833333e-01, 1.30208333e-01,
        3.25520833e-02, 8.13802083e-03, 8.13802083e-03, 3.25520833e-02],
       [3.33333333e+01, 8.33333333e+00, 0.00000000e+00, 3.25520833e-02,
        8.13802083e-03, 8.13802083e-03, 3.25520833e-02, 1.30208333e-01],
       [8.33333333e+00, 2.08333333e+00, 0.00000000e+00, 8.13802083e-03,
        8.13802083e-03, 3.25520833e-02, 1.30208333e-01, 5.20833333e-01],
       [2.08333333e+00, 5.20833333e-01, 0.00000000e+00, 2.03450521e-03,
        2.03450521e-03, 8.13802083e-03, 0.00000000e+00, 2.08333333e+00],
       [5.20833333e-01, 1.30208333e-01, 3.25520833e-02, 8.13802083e-03,
        2.03450521e-03, 2.03450521e-03, 0.00000000e+00, 8.33333333e+00],
       [1.30208333e-01, 3.25520833e-02, 8.13802083e-03, 2.03450521e-03,
        5.08626302e-04, 5.08626302e-04, 0.00000000e+00, 3.33333333e+01]])
```

شکل ۱۵ - خروجی کد تمرین سوم: نمایش مقادیر بهینه

نتیجه خروجی کد تمرین سوم: از لحاظ اجرا تفاوتی بین الگوریتم تکرار سیاست و الگوریتم تکرار مقدار شاهده نشده است و همانطور که گزارش شده است کاملا با هم برابر هستند.

تمرین چهارم

متن سوال چهارم: تمرین ۱۲-۳ کتاب را انجام دهید.

تابع مقدار حالت در s باید «میانگین» توابع مقدار عمل ممکن برای هر اقدام ممکن a باشد داریم:

$$V^{\pi}(s)=E_{\pi}\{ R_t \mid s_t = s \}$$

$$= \sum_a E_{\pi} \{R_t \mid s_t = s, s_t = a\} P \{a_t = a \mid s_t = s\}$$

$$= \sum_a E_{\pi} \{ R_t \mid s_t = s, a_t = a\} \pi(s, a).$$

با باز کردن عبارت expectation خواهیم داشت:

$$V^{\pi}(s) = \sum_a Q^{\pi}(s, a)\, \pi(s, a)$$

$$Q_{\pi}\big(s, \pi(s)\big) = V_{\pi}(s)$$