Q-Learning, SARSA, MC in Taxi Environment

Reinforcement Learning

Hamed Masoudi

9922102135

APRIL 29, 2023

Ferdowsi University of Mashhad
Faculty of Engineering

سوال اول:

ابتدا نیاز است یک Wrapper بنویسیم تا بتوانیم چهار تا اکشن خواسته پروژه را به محیط اضافه کنیم. Wrapper به صورت زیر نوشتم:

```
class ExtraActionWrapper(gym.Wrapper):
    def __init__(self, env):
        super().__init__(env)
        self.action_space = gym.spaces.Discrete(10)
        self.taxi_row, self.taxi_col, self.pass_idx, self.dest_idx =
self.env.env.decode(self.env.env.s)
        self.state = self.env.env.encode(self.taxi_row, self.taxi_col,
self.pass_idx, self.dest_idx)

    def step(self, action):
        # Map new actions to existing actions
        if action == 6:  # Up-East
            next_state, reward, done, info = self.env.step(1)
            if  self.state == next_state:
                return next_state, reward, done, info
            two_next_state, reward, done, info = self.env.step(2)
            if next_state == two_next_state:
                i,j,k,d=self.env.step(0)
                return i,j,k,d
            return two_next_state, reward, done, info
        elif action == 7:  # Up-West
            next_state, reward, done, info = self.env.step(1)
            if  self.state == next_state:
                return  next_state, reward, done, info
            two_next_state, reward, done, info = self.env.step(3)
            if next_state == two_next_state:
                i,j,k,d = self.env.step(0)
                return i,j,k,d
            return two_next_state, reward, done, info
        elif action == 8:  # Down-East
            next_state, reward, done, info = self.env.step(0)
            if  self.state == next_state:
                return next_state, reward, done, info
            two_next_state, reward, done, info = self.env.step(2)
            if next_state == two_next_state:
                i,j,k,d = self.env.step(1)
                return i,j,k,d
            return two_next_state, reward, done, info
        elif action == 9:  # Down-West
            next_state, reward, done, info = self.env.step(0)
```

```
            if  self.state == next_state:
                return next_state, reward, done, info
            two_next_state, reward, done, info = self.env.step(3)
            if next_state == two_next_state:
                i,j,k,d = self.env.step(1)
            return two_next_state, reward, done, info
        else:
            return self.env.step(action)

env = ExtraActionWrapper(env)
```

برای نوشتن Wrapper از کلاس استفاده می کنیم که از کلاس مادر gym.wrapper ارث بری می کند. در حقیقت این بخش کد متد step کتابخانه gym.env.step نسبت به مسئله مورد نیاز خودمان کاستومایز می کنیم.

```
env = ExtraActionWrapper(env)
```

این خط را قرار دادیم تا از env کاستومایز شده استفاده کنیم. در هر سه ماژولی که برای این تمرین نوشتم (Q-learning , MC First Visit, SARSA) از این env کاستومایز شده استفاده کردم.

سوال دوم:

به ازای این هایپر پارامتر ها الگوریتم MC First Visit را اجرا می کنیم:

```
num_episodes = 100000
epsilon = 0.1
discount_factor = 0.9
```

برای پیاده سازی الگوریتم MC First visit یک جدول Q پیاده سازی کردیم. این مسئله 500 تا State دارد همچنین برصورت عادی دارای 6 اکشن است ولی چون ما خواستیم 4 اکشن دیگه اضافه کنیم بنابرین مسئله ما دارای 10 اکشن است.

```
#0 -> Down
#1 -> Up
#2 -> Right
#3 -> left
#4 -> Pickup
#5 -> Drop
#6 -> Up-East
#7 -> Up- West
#8 -> Down- East
#9 -> Down- West
```

```
Q
```

✓ 0.0s

```
array([[  0.        ,   0.        ,   0.        , ...,   0.        ,
          0.        ,   0.        ],
       [ -8.64097433,  -7.74864162,  -8.29999987, ...,  -8.28824493,
         -9.2506725 ,  -9.10950571],
       [-10.6811157 , -10.62786437, -10.63699883, ..., -10.69825451,
        -10.63606113, -10.62759204],
       ...,
       [ -9.19180003, -11.12379763,  -9.8383113 , ..., -12.24073228,
        -12.40438828,  -1.66001345],
       [-10.8583481 , -12.58286836, -10.15072775, ..., -10.16550655,
        -10.12839147, -10.15947652],
       [-10.56666122, -10.01425943, -13.74117694, ..., -10.69958176,
          0.        ,  -9.65663162]])
```

```
Q.shape
```

✓ 0.0s

```
(500, 10)
```

شکل 1. جدول Q که دارای 500 سطر و 10 ستون است

```
df
```

✓ 0.0s

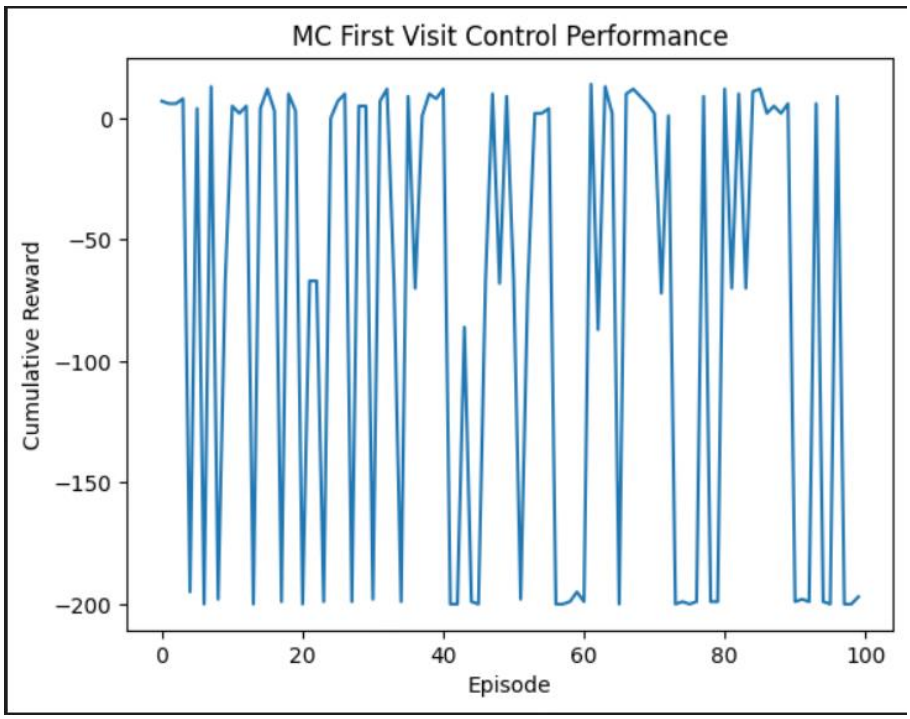
	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	-8.640974	-7.748642	-8.300000	-8.821370	3.309783	-16.688617	-8.956921	-8.288245	-9.250673	-9.109506
2	-10.681116	-10.627864	-10.636999	-10.626190	-10.629017	-19.520335	-10.677909	-10.698255	-10.636061	-10.627592
3	-10.708173	-10.659243	-10.675074	-10.656874	-10.658991	-19.626396	-10.672840	-10.694310	-10.660641	-10.666940
4	-11.035929	-11.010246	-8.663412	-10.965483	-19.968190	-19.727227	-11.033519	-11.043012	-11.147434	-10.988945
...	...	...	...	...	...	...	...	...	...	...
495	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
496	-2.290819	-1.551551	-2.735776	-3.768715	-11.904667	-12.208262	-3.288644	-5.282931	-0.312679	-3.620951
497	-9.191800	-11.123798	-9.838311	-10.162987	-20.719334	-19.279224	-10.951472	-12.240732	-12.404388	-1.660013
498	-10.858348	-12.582868	-10.150728	-10.157442	-21.005295	-20.183284	-10.582459	-10.165507	-10.128391	-10.159477
499	-10.566661	-10.014259	-13.741177	0.000000	-18.690968	0.000000	0.000000	-10.699582	0.000000	-9.656632

500 rows × 10 columns

شکل 2. نمایش بصورت Pandas Data Frame

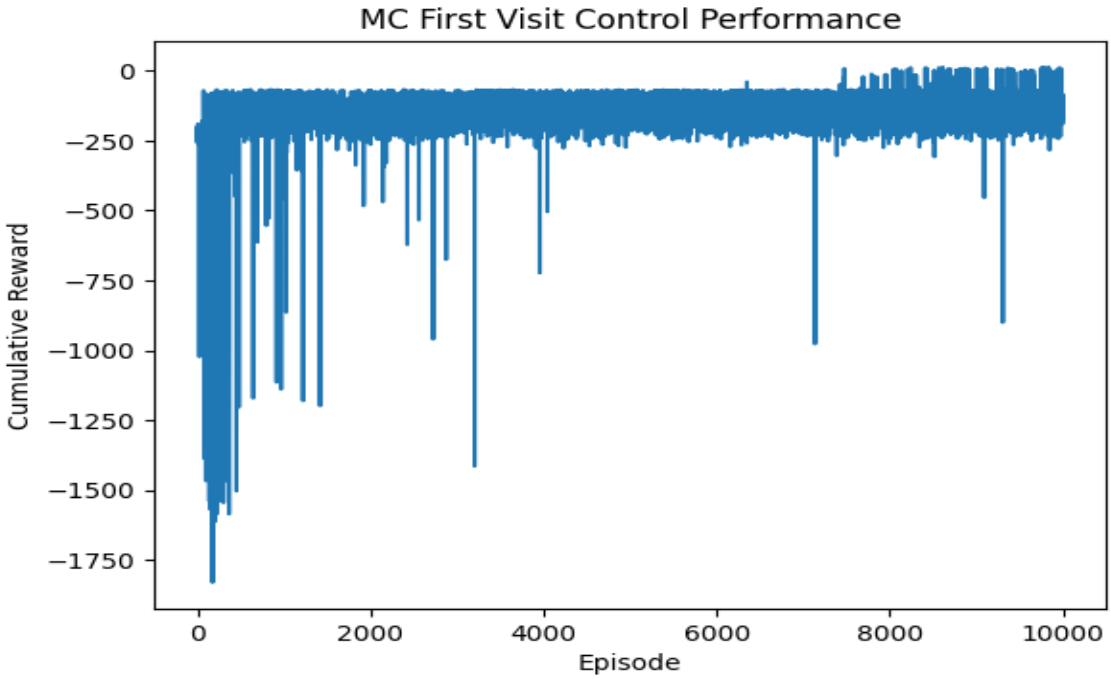
شکل 1 و شکل 2 برای زمانی هست که ما از گاما 0.9 استفاده کردیم با تعداد 100000 اپیزود.

نمودار پاداش تجمعی برای بعد از یادگیری مقادیر Q بصورت زیر است:



شکل 3. نمودار پاداش تجمعی MC به ازای گاما 0.9

حال نمودار پاداش تجمعی به ازای گاما 0.9 در حال یادگیری را رسم می کنیم یعنی هر اپیزود از یادگیری پاداش تجمعی را محاسبه می کنیم:



شکل 4. پاداش تجمعی عامل برای MC در حال یادگیری با گاما 0.9

مشخص است که الگوریتم همگرا شده است.

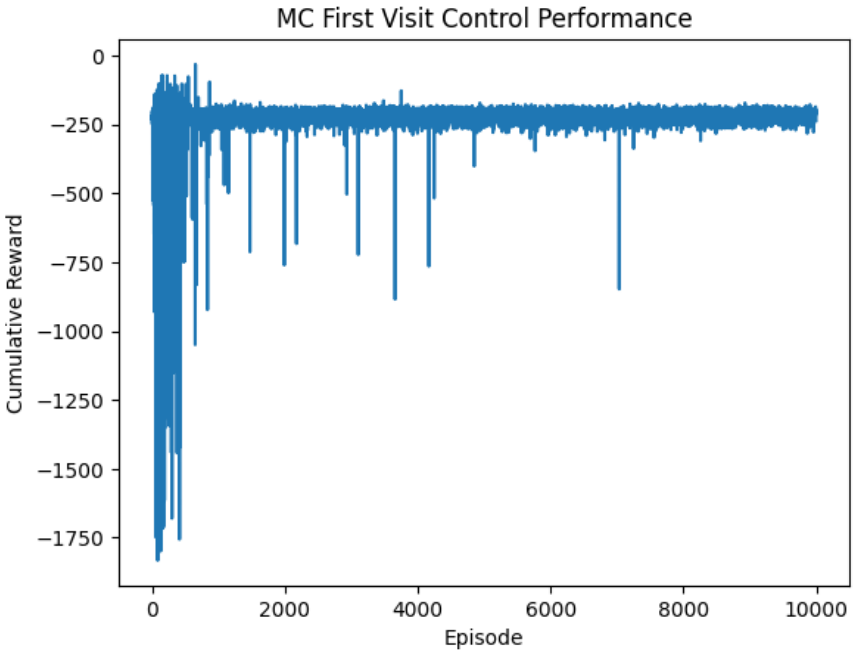
حال مقادیر Q و پاداش تجمعی برای گاما 0 را بررسی می کنیم.

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	-1.0	-1.0	-1.0	-1.0	-1.0	-10.0	-1.0	-1.0	-1.0	-1.0
2	-1.0	-1.0	-1.0	-1.0	-1.0	-10.0	-1.0	-1.0	-1.0	-1.0
3	-1.0	-1.0	-1.0	-1.0	-1.0	-10.0	-1.0	-1.0	-1.0	-1.0
4	-1.0	-1.0	-1.0	-1.0	-10.0	-10.0	-1.0	-1.0	-1.0	-1.0
...	...	...	...	...	...	...	...	...	...	...
495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
496	-1.0	-1.0	-1.0	-1.0	-10.0	-10.0	-1.0	-1.0	-1.0	-1.0
497	-1.0	-1.0	-1.0	-1.0	-10.0	-10.0	-1.0	-1.0	-1.0	-1.0
498	-1.0	-1.0	-1.0	-1.0	-10.0	-10.0	-1.0	-1.0	-1.0	-1.0
499	-1.0	-1.0	-1.0	-1.0	-10.0	-10.0	-1.0	-1.0	-1.0	-1.0

500 rows × 10 columns

شکل 5. نمایش ارزش بصورت Data Frame Pandas برای گاما 0

نمودار پاداش تجمعی به ازای گاما 0 برای MC در حال یادگیری:



شکل 6. پاداش تجمعی عامل برای MC در حال یادگیری با گاما 0

مشخص است که نباید گاما را برابر 0 قرار دهیم به شدت عملکرد کارگزار افت می کند. مقایسه شکل 6 و 4 نشان می دهد بطور متوسط با گاما 0.9 عامل بهتر عمل کرده است.

وقتی گاما برابر با 0 باشد عامل به آینده تصمیم فکری نمی کند در حقیقت Return ما فقط پاداش حال حاضر می شود این باعث می شود که عامل تصمیم گیری اشتباه انجام بدهد.

سوال سوم:

به ازای هایپر پارامتر های زیر الگوریتم Q-Learning را اجرا می کنیم:

```
alpha = 0.1
gamma = 0.9
epsilon = 0.1
num_episodes = 100000
```

در این الگوریتم alpha همان learning rate است.

همچنین برای اینکه به مقادیر صحیح Q برسیم به ازای 100000 اپیزود یادگیری را انجام دادیم.

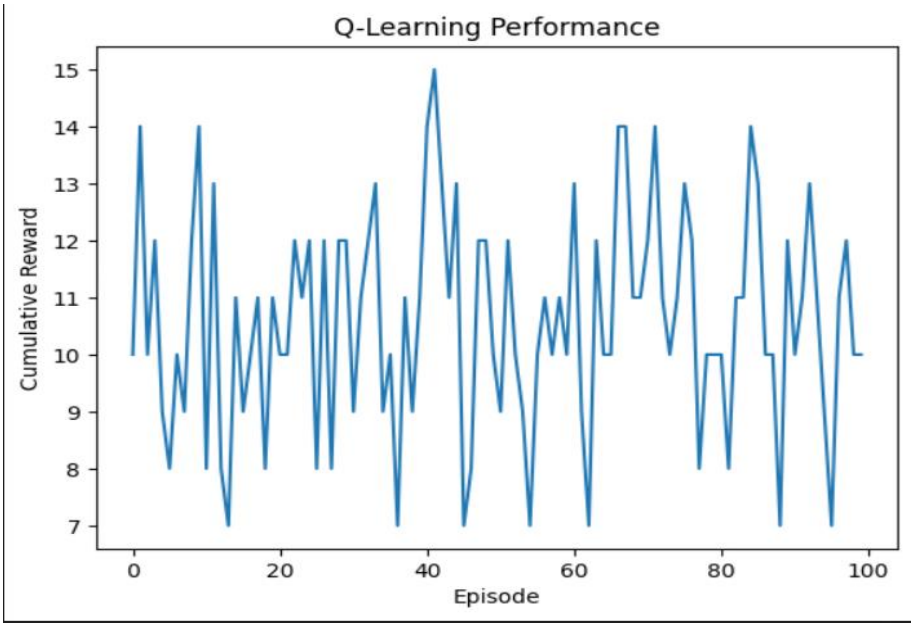
جدول ارزش ها الگوریتم Q Learning به ازای گاما 0.9 بصورت زیر است:

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	4.348705	5.942978	4.348786	5.942666	7.714700	-3.056888	4.348655	4.347857	4.347715	4.348767
2	1.365996	4.249027	1.407328	3.741438	7.443961	-3.826572	2.843034	2.601795	2.678314	3.196006
3	2.849671	4.290461	2.857810	4.312172	5.843881	-4.727159	2.858721	2.866092	2.833669	2.859544
4	-2.791526	-2.553582	-2.525825	-2.367489	-4.897358	-6.545658	-2.188176	-2.524673	0.460353	-2.171028
...	...	...	...	...	...	...	...	...	...	...
495	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
496	1.045261	1.578349	1.046169	0.918309	-1.390788	-1.707652	-0.095054	7.714700	-0.157274	-0.708368
497	0.326754	3.120026	0.327800	2.461341	-1.000000	-1.709336	1.666187	1.598601	1.929846	7.716800
498	-1.085626	-3.699113	-2.376370	-1.941289	-8.539694	-7.816767	-3.515048	5.463739	-1.629626	-1.736417
499	6.877288	14.094990	5.709429	7.419575	0.956094	1.979616	6.331467	7.930725	8.963277	12.844003

500 rows × 10 columns

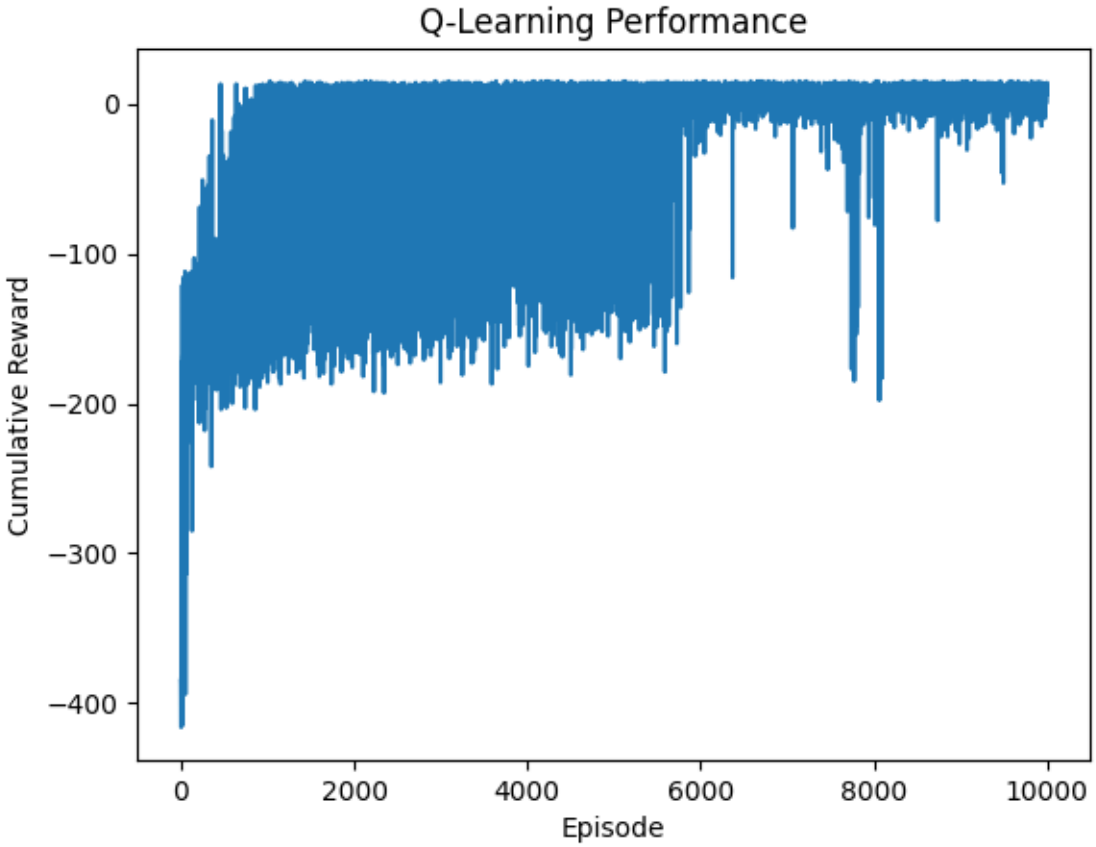
شکل 7. جدول Q برای Q learning با گاما 0.9

نمودار پاداش تجمعی بصورت زیر است:



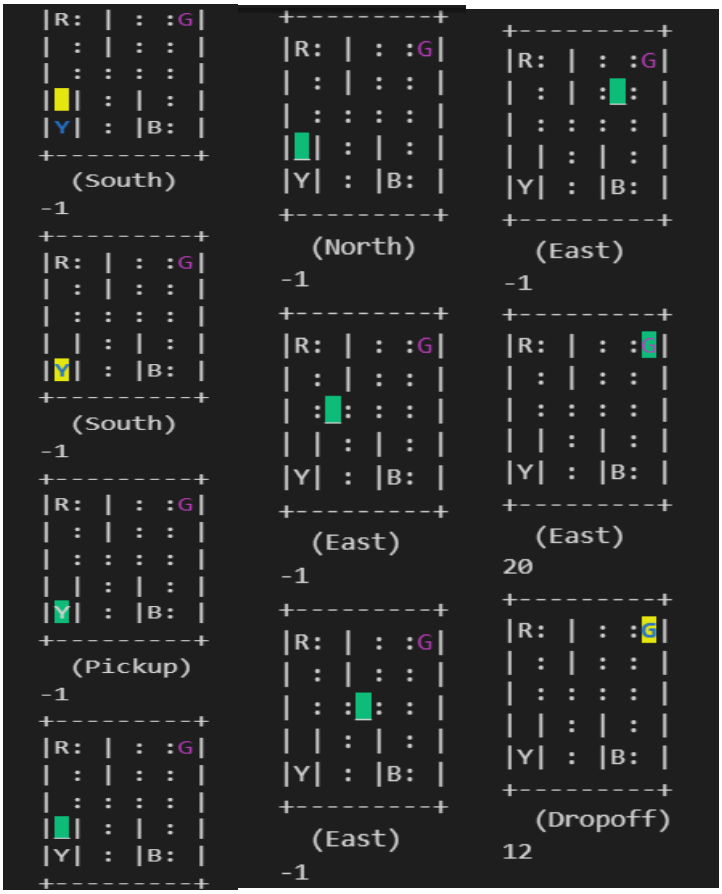
شکل 8.نمودار پاداش تجمعی q learning به ازای گاما 0.9

نمودار پاداش تجمعی در حال یادگیری بصورت زیر است:



شکل 9. پاداش تجمعی عامل برای Q learning در حال یادگیری با گاما 0.9

واضیح است که عملکرد Q learning بسیار از MC بهتر است. تمام 100 اپیزود تست را الگوریتم توانسته به خوبی تمام کند. بعضی از اپیزود ها که سخت تر بودن عامل ریوارد کمتری بدست اورده زیرا بیشتر مجبور بوده در محیط حرکت کند و هر حرکت عامل در محیط ریوارد منفی می گیرد. حال قابل توجه هست که حرکت عامل را برای یک اپیزود بررسی کنیم:



مشخص است که عامل در این اپیزود چقدر بهینه عمل کرده است. حتی یک اشتباه هم نداشته است و تمام تصمیماتی که عامل در این اپیزود گرفته است درست است.

این نشان می دهد الگوریتم Q learning در عمل هم بسیار کار آمد است. اگر به شکل9 دقت کنیم متوجه می شویم که الگوریتم همگرا شده است و بطور متوسط نسبت به MC پاداش بیشتری در محیط کسب می کند

نتایجی که تا الان برای Q learning بررسی کردیم برای زمانی بود که گاما برابر با 0.9 بود.

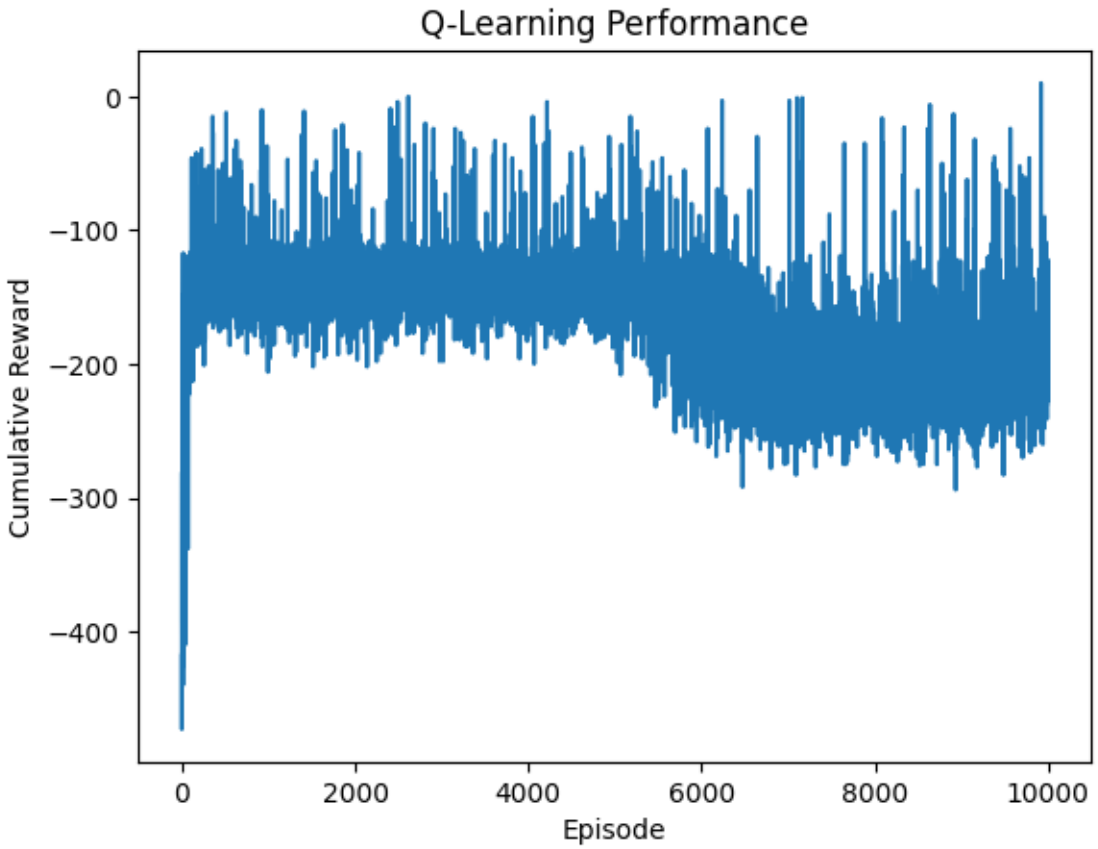
انتظار داریم زمانی که گاما دقیقا برابر با 0 باشد عامل به خوبی نتواند در محیط رفتار کنید.

حال نتایج را برای زمانی که گاما 0 باشد بررسی می کنیم:

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-9.202336	-1.000000	-1.000000	-1.000000	-1.000000
2	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-8.784233	-1.000000	-1.000000	-1.000000	-1.000000
3	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-8.649148	-1.000000	-1.000000	-1.000000	-1.000000
4	-1.000000	-1.000000	-1.000000	-1.000000	-9.113706	-9.202336	-1.000000	-1.000000	-1.000000	-1.000000
...	...	...	...	...	...	...	...	...	...	...
495	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
496	-1.000000	-1.000000	-1.000000	-1.000000	-10.000000	-10.000000	-1.000000	-1.000000	-1.000000	-1.000000
497	-1.000000	-1.000000	-1.000000	-1.000000	-10.000000	-10.000000	-1.000000	-1.000000	-1.000000	-1.000000
498	-1.000000	-1.000000	-1.000000	-1.000000	-10.000000	-10.000000	-1.000000	-1.000000	-1.000000	-1.000000
499	-0.972187	-0.972187	-0.972187	-0.972187	-3.439000	-2.710000	-0.972187	-0.972187	-0.972187	-0.969097

500 rows × 10 columns

شکل 10.جدول Q برای Q learning با گاما 0



شکل 11. پاداش تجمعی عامل برای Q learning در حال یادگیری با گاما 0

مشخص است که عامل خیلی از اپیزود ها را حتی نتوانسته تموم کند . در حقیقت عامل نتوانسته با گاما برابر با 0 به جدول Q مناسبی برسد. بصورت متوسط هم مشخص است که عملکرد عامل با گاما 0 خیلی نویزی تر و ضعیف تر است نسبت به گاما 0.9 دلیل این امر را در MC توضیح دادیم.

سوال چهارم:

به ازای هایپر پارامتر های زیر الگوریتم SARSA را اجرا می کنیم:

```
alpha = 0.1
gamma = 0.9
epsilon = 0.1
num_episodes = 100000
```

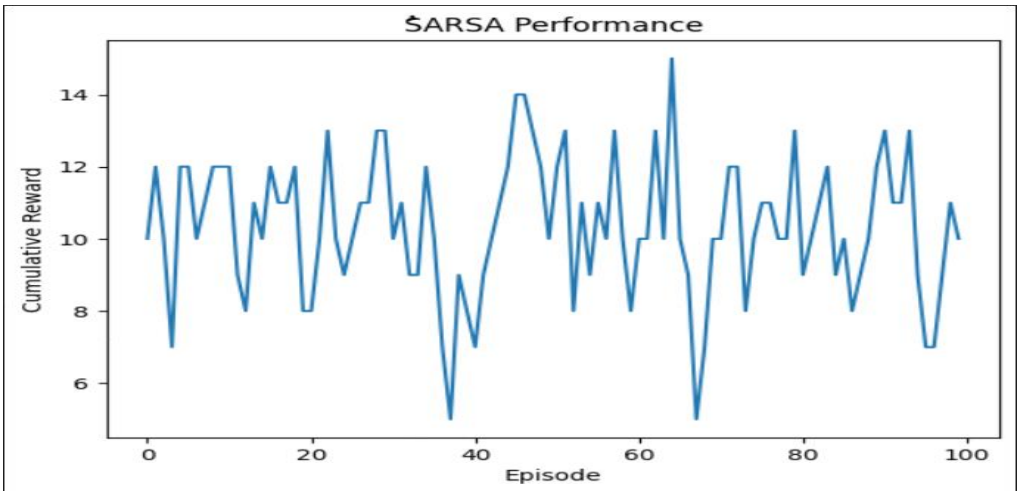
جدول Q بصورت زیر است:

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	1.465138	3.900244	1.829196	3.894709	5.862704	-4.638291	1.768995	2.345307	2.271379	2.367357
2	-6.258567	-4.209470	-4.707447	-4.916317	2.944554	-13.602810	-6.809043	-6.202317	-8.718597	-6.157260
3	-0.820559	1.991418	0.351847	1.877443	2.996760	-7.928711	0.573599	-0.650566	0.924252	-0.624582
4	-3.631593	-3.503295	-3.468666	-3.380070	-8.057557	-6.482502	-3.740566	-3.812882	-2.013124	-3.569014
...	...	...	...	...	...	...	...	...	...	...
495	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
496	-0.497086	-0.672942	0.053598	-1.143303	-3.260451	-2.387879	-0.221770	-0.387309	3.853749	-0.874981
497	0.435130	2.048949	1.765869	0.757284	-1.454854	-2.181845	0.700456	8.437012	0.778309	2.244473
498	-9.005368	-6.047908	-9.041186	-9.394159	-14.657492	-14.616892	-9.241235	-9.606782	-9.385733	-8.979764
499	5.849601	6.590525	5.834935	15.358290	2.768898	3.404639	6.310714	5.852225	5.831022	5.870976

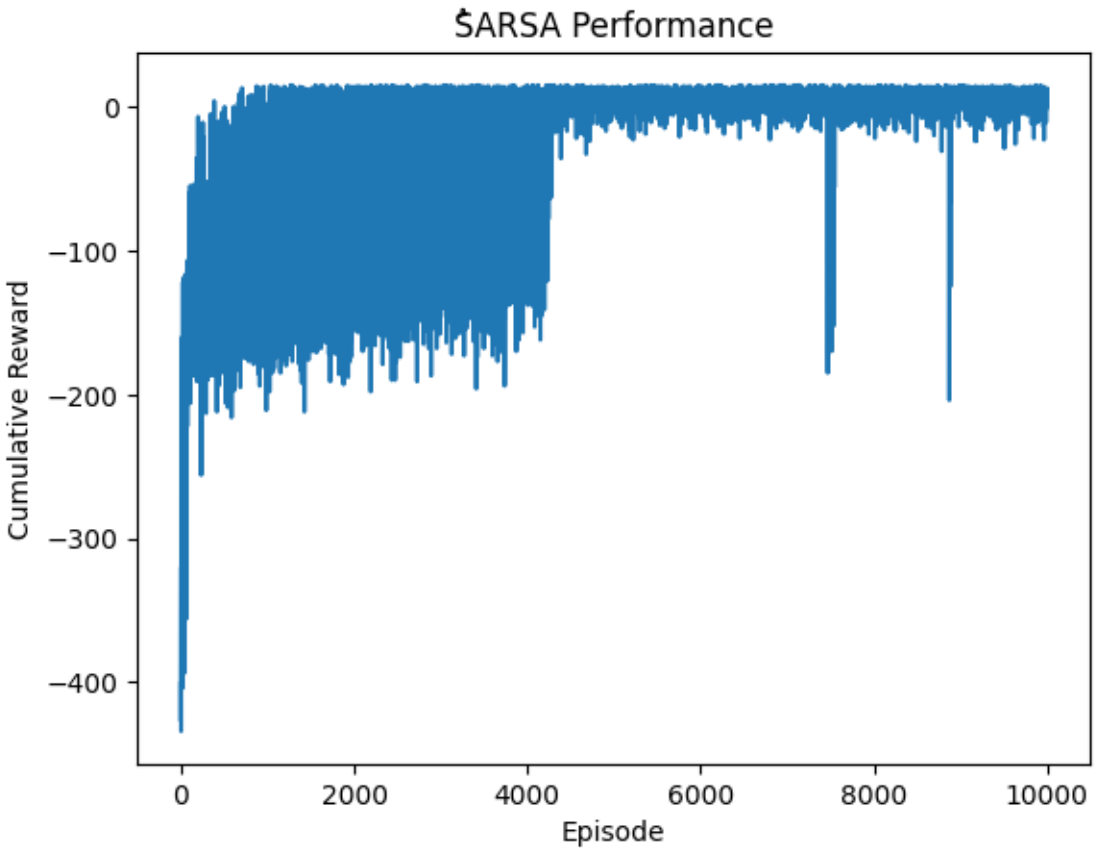
500 rows × 10 columns

شکل 12. جدول Q برای SARSA با گاما 0.9

نمودار پاداش تجمعی بصورت زیر است:



شکل 13. نمودار پاداش تجمعی SARSA بعد از یادگیری Q به ازای گاما 0.9



شکل 14.پاداش تجمعی عامل برای SARSA در حال یادگیری Q با گاما 0.9

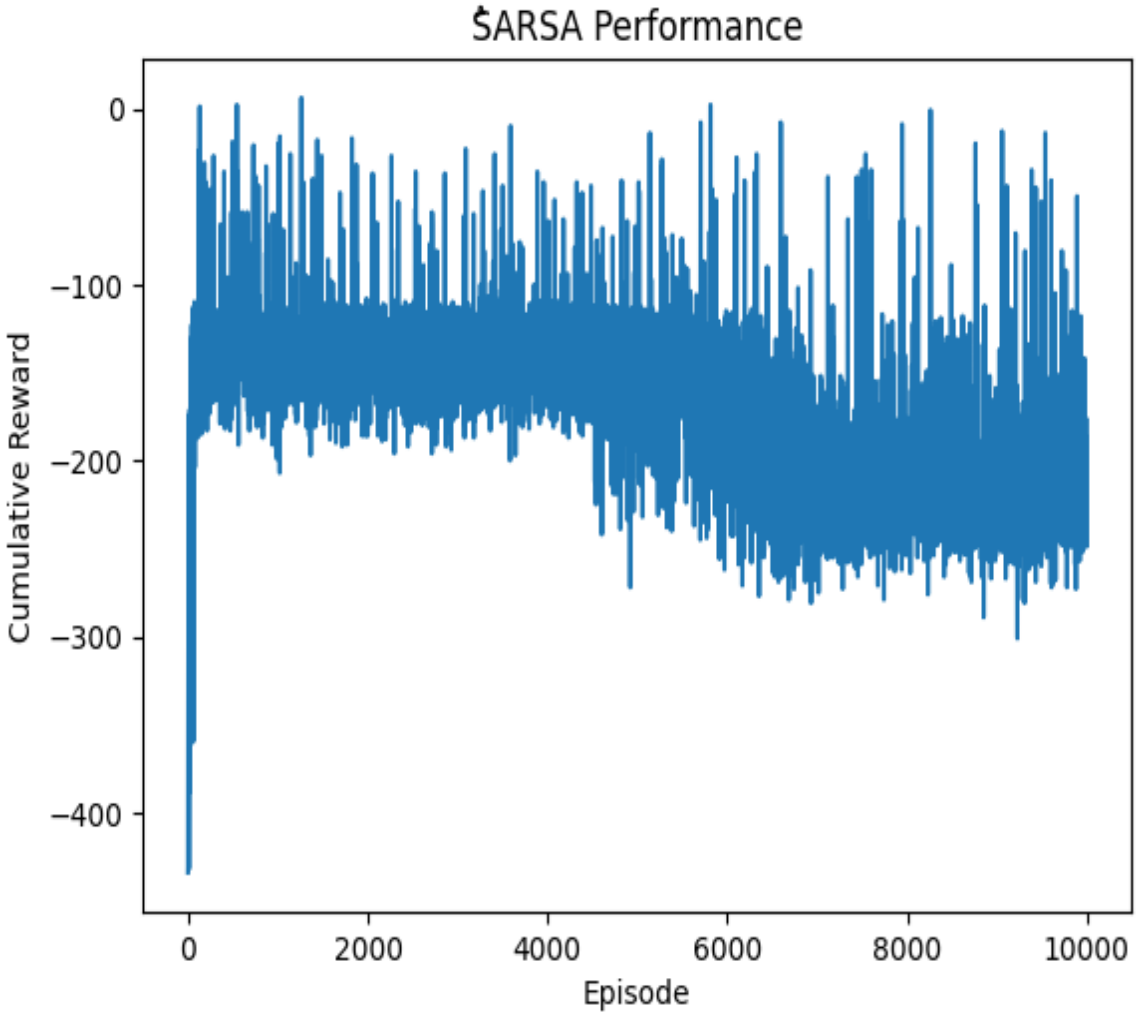
تقریبا میشه گفت عملکرد Q learning بهتر بود اما خیلی عملکرد نزدیکی به هم دارند در گاما 0.9.

حال برای گاما 0 بررسی می کنیم. جدول Q برای SARSA با گاما 0 بصورت زیر است:

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-9.576088	-1.000000	-1.000000	-1.000000	-1.000000
2	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-8.649148	-1.000000	-1.000000	-1.000000	-1.000000
3	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-8.784233	-1.000000	-1.000000	-1.000000	-1.000000
4	-1.000000	-1.000000	-1.000000	-1.000000	-8.499054	-9.576088	-1.000000	-1.000000	-1.000000	-1.000000
...	...	...	...	...	...	...	...	...	...	...
495	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
496	-1.000000	-1.000000	-1.000000	-1.000000	-10.000000	-10.000000	-1.000000	-1.000000	-1.000000	-1.000000
497	-1.000000	-1.000000	-1.000000	-1.000000	-10.000000	-10.000000	-1.000000	-1.000000	-1.000000	-1.000000
498	-1.000000	-1.000000	-1.000000	-1.000000	-10.000000	-10.000000	-1.000000	-1.000000	-1.000000	-1.000000
499	-0.972187	-0.972187	-0.974968	-0.972187	-2.710000	-2.710000	-0.972187	-0.972187	-0.972187	-0.972187

500 rows × 10 columns

شکل 15.جدول Q برای SARSA با گاما 0



شکل 16 . نمودار پاداش تجمعی برای SARSA در زمان یادگیری Q گاما 0

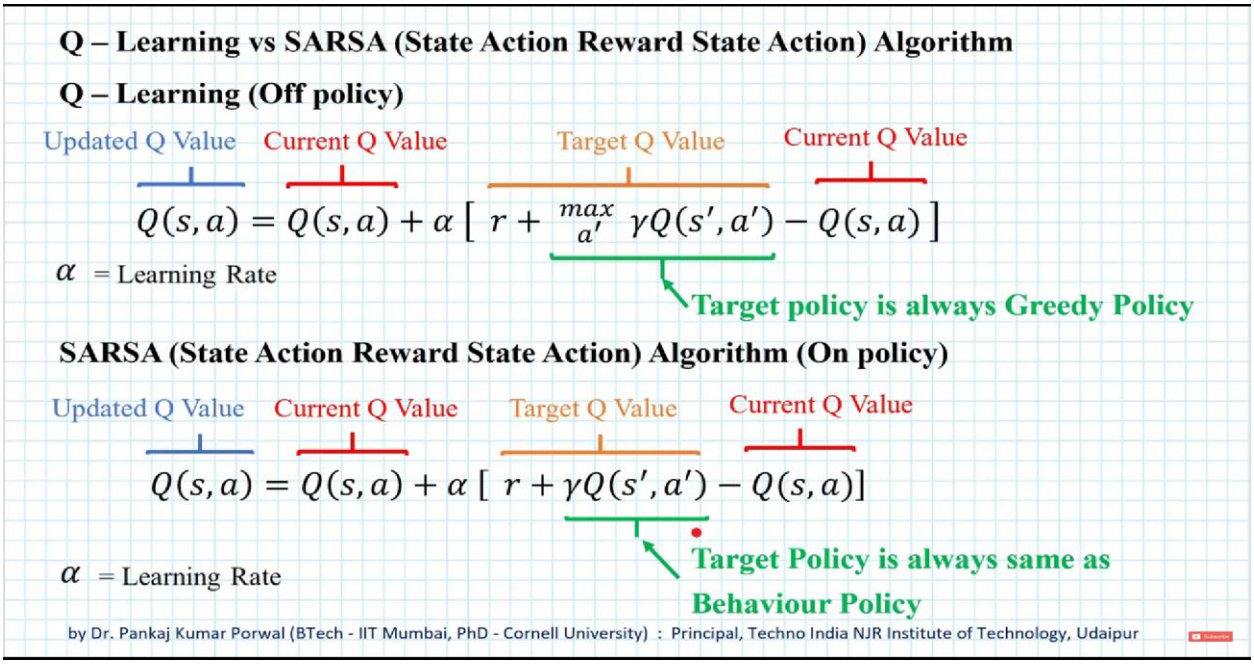
مشخص است که عملکرد هیچ کدام از الگوریتم با گاما 0 جالب نیست بخاطر دلیل هایی که در قبل ذکر کردیم.

در این حالت هم مشخص است که نمودار پاداش تجمعی بصورت میانگین در حالت گاما 0 کمتر از گاما 0.9 است

نتیجه می گیریم که قرار دادن گاما با 0 در مسائل RL عموما ایده خوبی نیست.

سوال چهارم:

الگوریتم SARSA و Q learning هر دو الگوریتم کنترلی TD هستند ولی تارگت آن ها فرق دارد. در حقیقت Q learning یک الگوریتم Off policy کنترل است بطوری که سیاستی که اپدیت می شود با سیاستی که ما در آن در محیط رفتار می کنیم متفاوت است اما SARSA یک الگوریتم On policy کنترل است.



شکل 17. تفاوت Q learning و SARSA

مشخص است تارگت سیاست الگوریتم Q learning حریصانه است و تفاوت دارد با سیاست حرکت در محیط که اپسیلون گریدی می تواند باشد.

اما الگوریتم SARSA همیشه تارگت سیاست با سیاستی که در محیط رفتار می کند یکی است. این باعث می شود که SARSA یک الگوریتم ON Policy باشد.