بخش تحلیلی:

سوال اول:

الگوریتم Q-learning، یک الگوریتم off-policy است .این الگوریتم در طول یادگیری سعی در یادگرفتن سیاست بهینه دارد چون آپدیت کردن action-value function یا همان Q با توجه به بهترین اکشن استیت بعد انجام می‌شود.

ولی الگوریتم SARSA که یک روش on-policy است. همان سیاستی را بهینه می‌کند که با آن زندگی می‌کند و Q را مستقیما از تعامل با محیط با توجه به سیاست یاد می‌گیرد(در این مثال سیاست e-greedy).

در این مثال مسیر کوتاه تر توسط الگوریتم Q-learning و مسیر طولانی تر(امن تر) توسط SARSA یاد گرفته می‌شود چون sarsa در طول یادگیری تماما با سیاست پیش می‌رود و از آنجا که اپسیلون نیز برابر 0.1 قرار داده شده بنابراین در طول یادگیری از ریواردهای منفی بیشتر فاصه می‌گیرد.
و Q-learning با توجه به توضیح گفته شده می‌خواهد سیاست بهینه را یاد بگیرد که مسیر نزدیک به مربع های قرمز است. ولی همچنان، عامل روی مربع های قرمز می‌رود چون انتخاب اکشن با توجه به سیاست e-greedy صورت می‌گیرد. به همین دلیل عمکلرد sarsa در مجموع بهتر از Q-learning است.

سوال دوم:

الف)
مقدار بهینه ارزش هر استیت، مجموع ریواردی است که از آن استیت تا استیت ترمینال تحت سیاست بهینه داریم:

حالت اول: rs=1:

در این حالت، ارزش سیاست های نزدیک تر به خانه 12 کمتر از خانه های دورتر می‌شود و عامل در طول یادگیری بیشتر در خانه‌های سفید می‌چرخد چون ارزش بیشتری دارند و به همین دلیل خانه هدف پیدا نمی‌شود.

حالت دوم: rs=0:
در این حالت ارزش همه استیت ها برابر می‌شود و عامل ترجیحی بر انتقال به استیت دیگر ندارد و به همین به خانه هدف نمی‌رسد.

حالت دوم: rs=-1:
در اینجا ارزش خانه های نزدیک تر به خانه هدف بیشتر از خانه های دورتر هستند و چون ریوارد حرکت بین مربع های سفید منفی است، عامل به خانه هدف همگرا می‌شود.

ارزش هر استیت:

1 1	2 2	3 3	4 4
5	6 3	7 4	8 5
9 3	10 4	11 5	12
13 2	14 1	15 0	16 -1

ب)

همانطور که در بخش قبل دیدیم در صورتی که rs مثبت انتخاب شود، عامل بین خانه های سفید می‌چرخد چون سیاست بهینه حرکت بین خانه های سفید است. ارزش هر استیت مطابق بخش قبل محاسبه می‌شود.

ج) از بخش قبل: rs=1:

در این حالت، ارزش هر استیت به صورت زیر محاسبه می‌شود:

$$R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$$

چون R=1:

$$1 + \gamma + \gamma^2 + \gamma^3 + \cdots$$

عبارت فوق را می توان به صورت زیر نوشت:

$$\frac{1}{1-\gamma}$$

در عبارت فوق اگر مقدار گاما بزرگ باشد ارزش استیت s، به جز خانه سبز بی نهایت می‌شود و عامل خانه سبز را پیدا نمیکند و بین خانه های سفید می‌چرخد.
اگر مقدار گاما کوچک باشد، ارزش استیت ها یک می‌شود به جز خانه سبز و عامل خانه سبز را پیدا می‌کند.

بنابراین سیاست بهینه تغییر می‌کند و این به مقدار گاما بستگی دارد.

د)

ارزش جدید را به صورت زیر تشکیل می‌دهیم. با فرض نامحدود بود horizon داریم:

$$V_{old}^{\pi} = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$$

$$V_{new}^{\pi} = (R_t + c) + \gamma (R_{t+1} + c) + \gamma^2 (R_{t+2} + c) + \cdots$$

$$= \underbrace{(R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots)}_{V_{old}^{\pi}} + c(1 + \gamma + \gamma^2 + \cdots)$$

$$= V_{old}^{\pi} + \frac{c}{1-\gamma}$$

در عبارت بالا، اگر مقدار گاما بزرگ انتخاب شود عبارت کسری به بی نهایت میل می‌کند و ارزش استیت s در حالتی که یک ریوارد ثابت اضافه می‌کنیم، بی نهایت می‌شود.

اگر نیز گاما کوچک انتخاب شود ارزش جدید برابر با ارزش قبلی می‌شود.

سوال سوم:

الگوریتم expected sarsa بر خلاف sarsa به جای sample گرفتن از state action values با توجه به سیاست بین تمام اکشن ها میانگین حساب می‌کند.

همانطور که انتظار می‌رود، E-SARSA محاسبات بیشتری از SARSA دارد ولی با توجه به اینکه بین تمام اکشن ها میانگین می‌گیرد واریانس حاصل از sampling در الگوریتم sarsa را ندارد.

اما اختلاف عملکرد این دو روش در محیط ها با stochasticity مختلف متفاوت است.

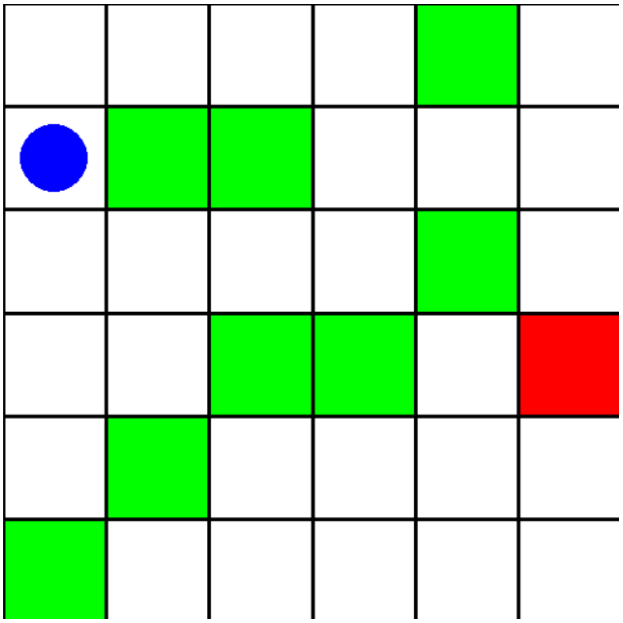
اگر محیط خیلی stochastic باشد این دو روش تفاوت زیادی ندارد چون randomness محیط روی این الگوریتم نیز تاثیر می‌گذارد.

اگر سیاست e-optimal باشد، با توجه به مقداری که برای پارامتر سیاست(مثلا اپسیلون) در نظر می‌گیریم اختلاف عملکرد دو روش متفاوت است ولی به طور کلی روش E-SARSA به همان دلیل ذکر شده در بالا، بهتر از SARSA است. اگر سیاست randomness بیشتری داشته باشد روش E-SARSA بهتر از SARSA عمل می‌کند.

ولی اگر سیاست، deterministic باشد، عملکرد این دو روش خیلی تفاوتی نخواهد کرد.

بخش پیاده سازی:

نوع قرار گیری عامل، هدف و درخت ها به صورت زیر شد با SEED=422



.1
در این بخش مقدار اپسیلون را با منطق زیر کاهشی در نظر گرفتم:

$$\frac{100}{100+i}$$

که در عبارت بالا i، شماره اپیزود است. در اپیزود اول مقدار اپسیلون 1 و به مرور در اپیزودهای بعد کم می‌شود. این کاهش به طوری است که به مرور از exploration به explotation برسیم.

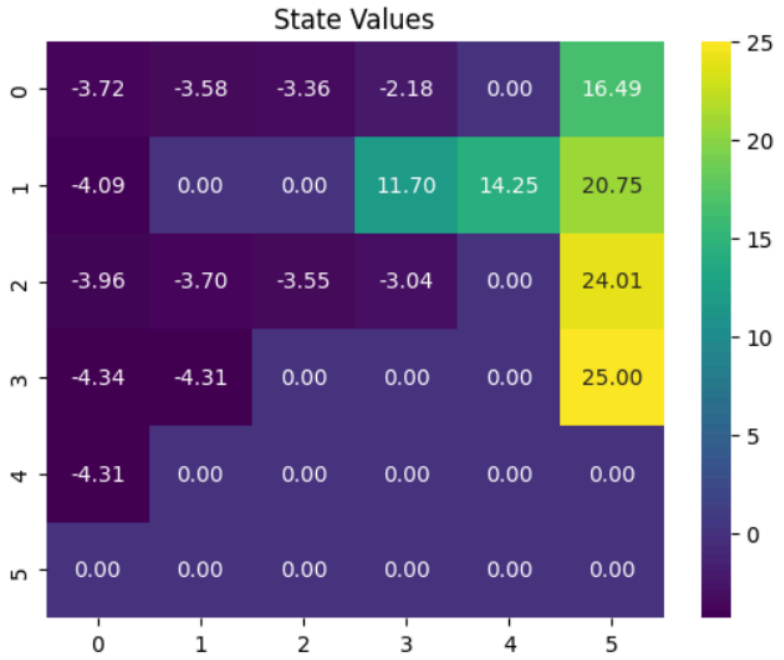
مقدار پارامتر یادگیری را نیز به شکل زیر کاهش دادم:(عامل در این حالت عملکرد بهتری داشت)

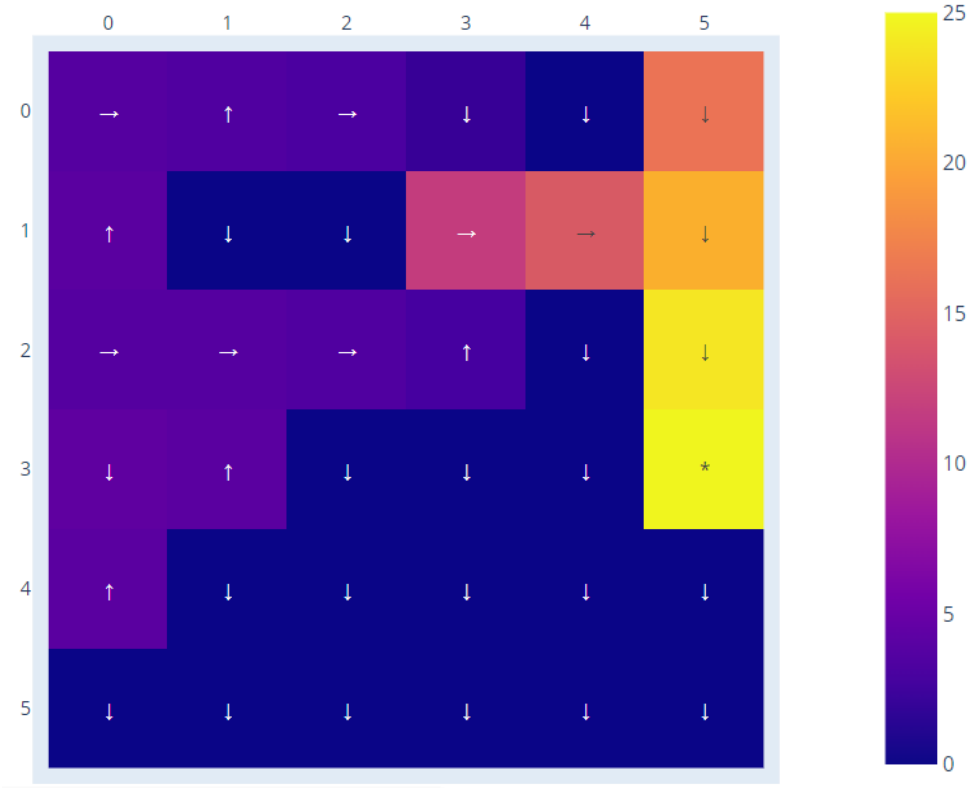
$$\frac{10}{100+i}$$

روش Q_learning:

1) lr=0.1

نمودارهای ارزش حالت ها و مسیر طی شده:



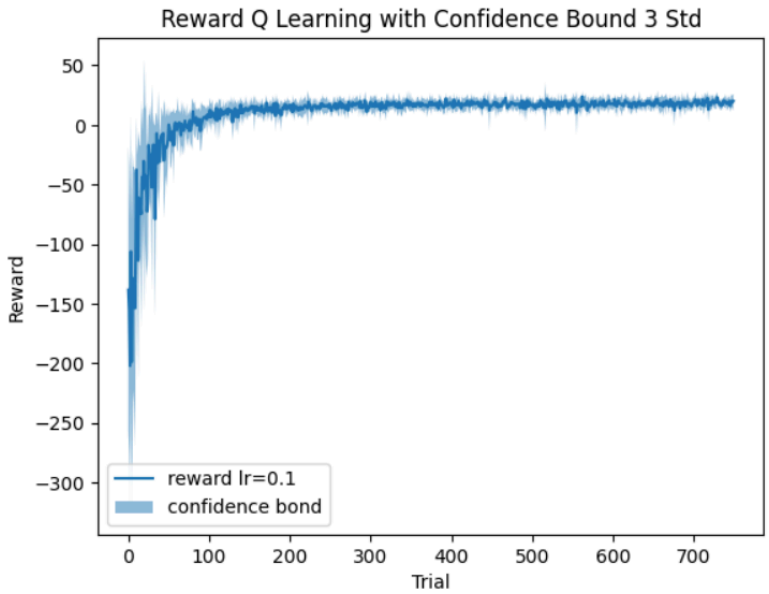


در شکل ها بالا می‌بینیم که عامل 2

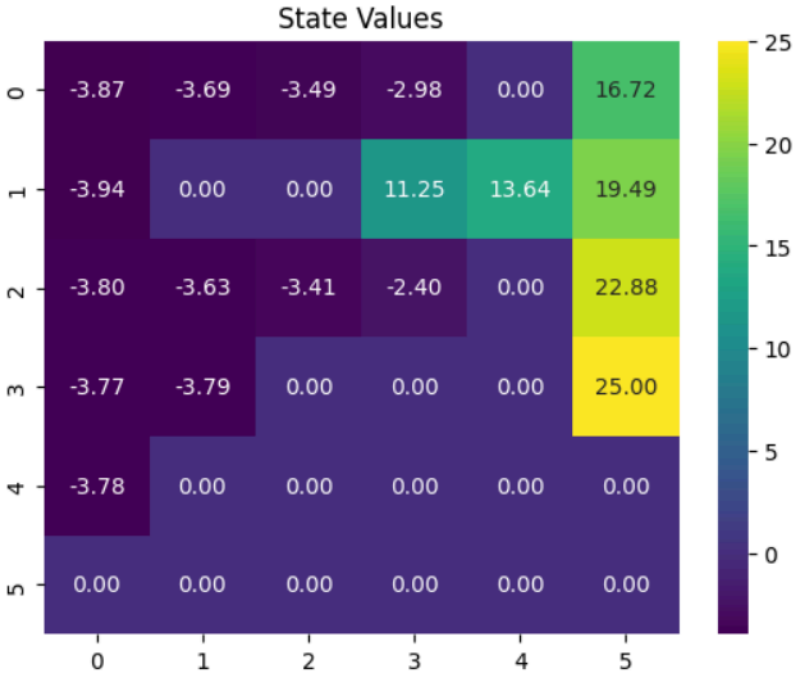
مسیر پیدا کرده است که یکی از آنها در تمام استیت ها درست است و در نمودار ارزش، نیز مقدار ارزش استیت های مسیر درست افزایش یافته اند.

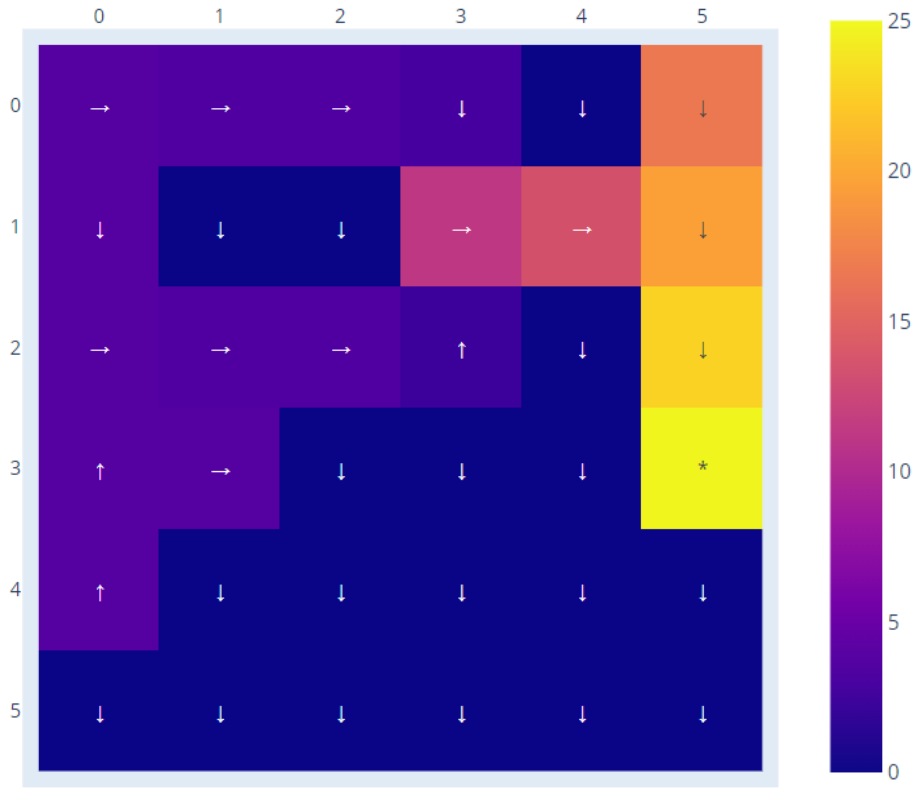
چون ارزش استیت ترمینال تغییر نمی‌کرد به صورت دستی آن را 25 قرار دادم.

نمودار میانگین پاداش ها به ازای 10 بار اجرا روی 750 اپیزود:

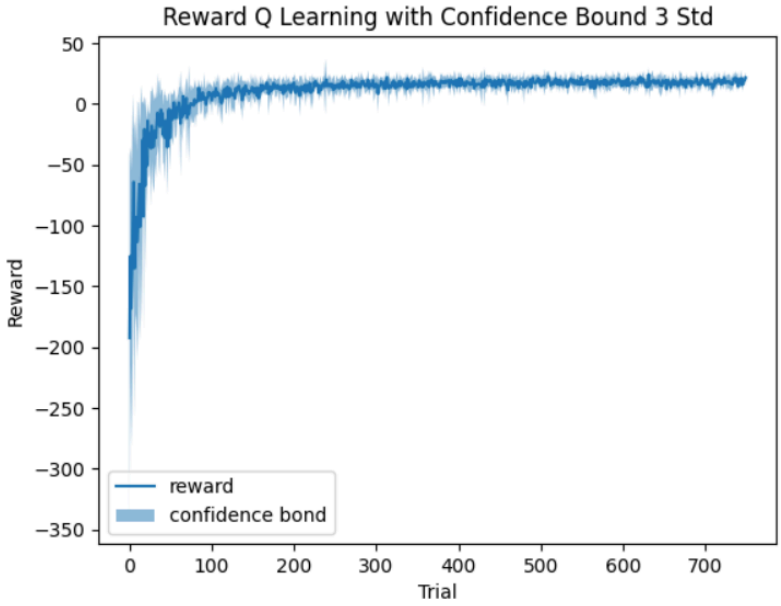


2) Decreasing learning rate:



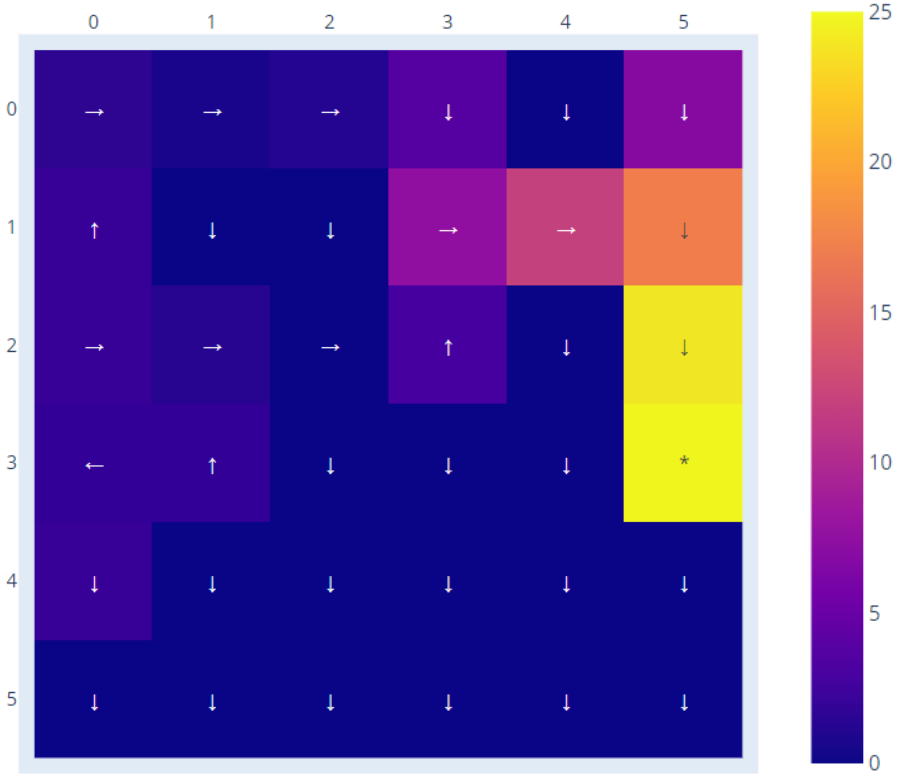


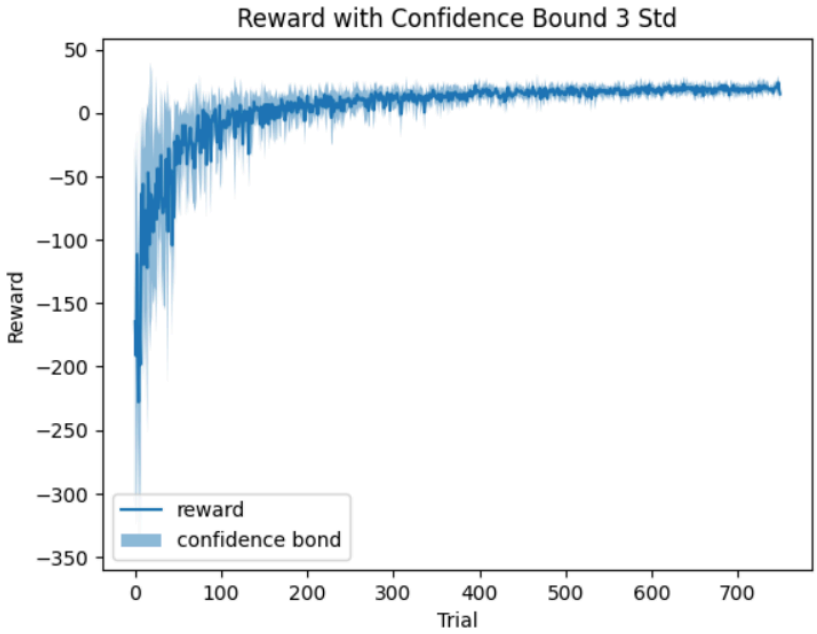
می بینیم که در این حالت مسیر یادگیری شده توسط عامل متفاوت با حالت قبل است.



مشاهده می‌شود کمی از مقدار واریانس پاداش در این حالت کم تر شده است.

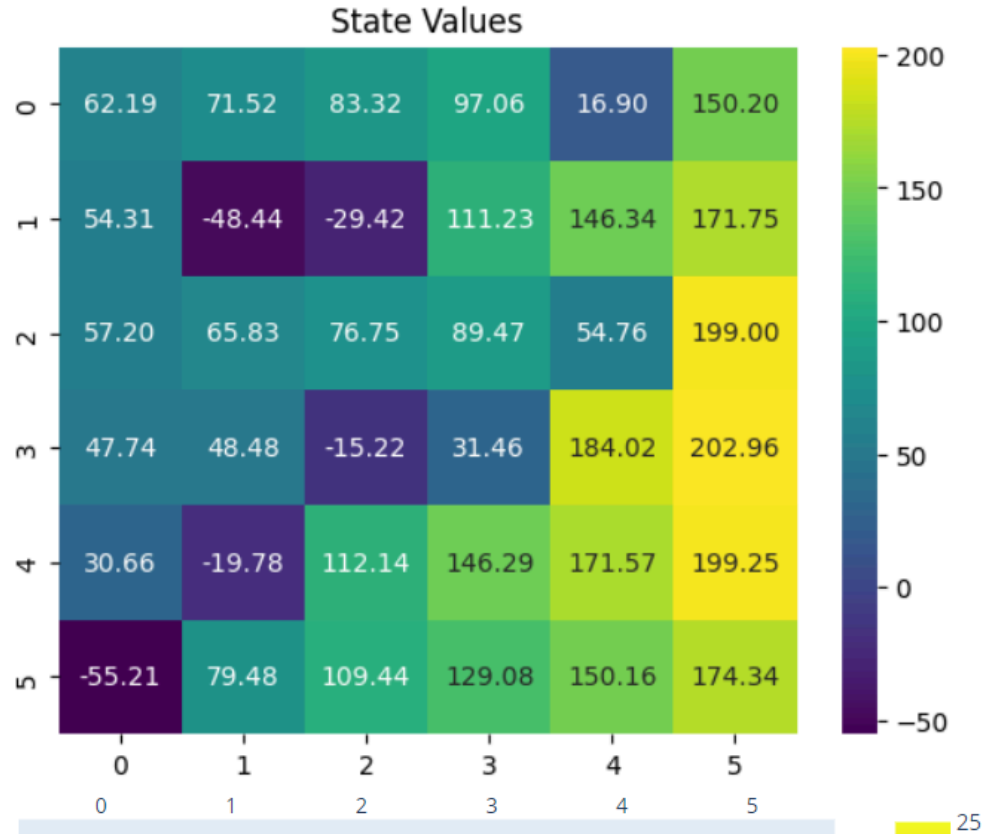
2) روش SARSA
به ازای lr ثابت:

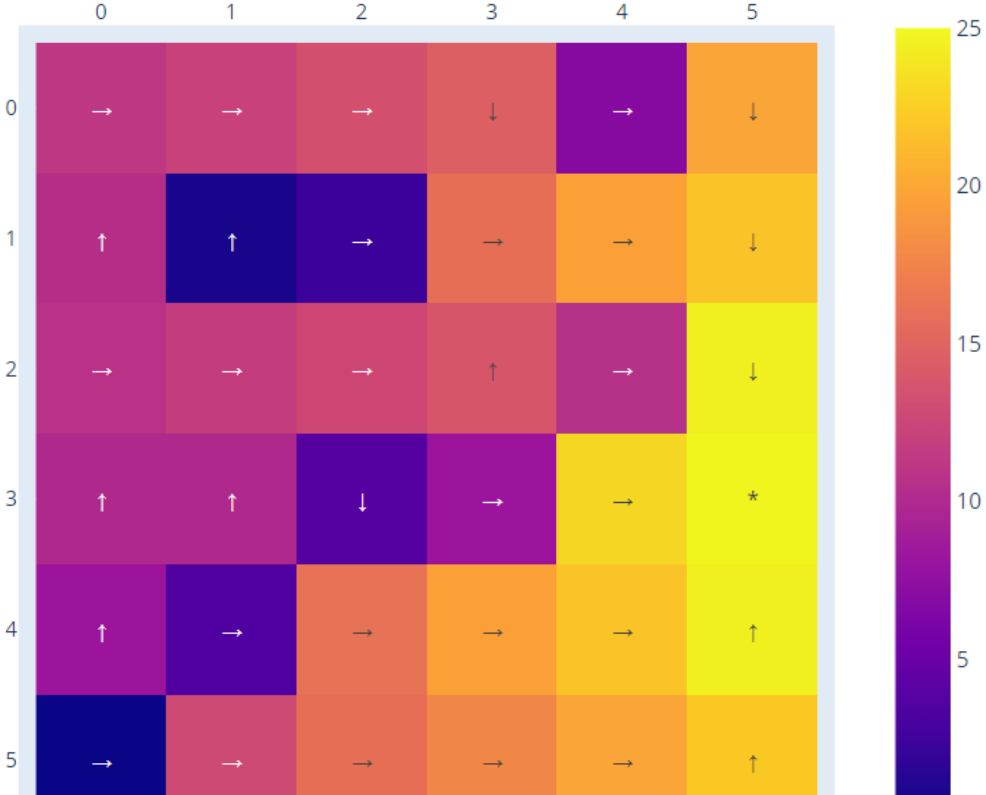




طبق شکل بالا زمان همگرایی SARSA از Q-learning بیشتر است و همچنین هر دو مسیر تشخیص داده شدند.

3,2) policy iteration





در این بخش، با استفاده از کد موجود در هندزآن 2 مقدار state_action_value ها تخمین زده شد.

برای استیت هایی که درخت دارند مقدار Q را 100 تا کم کردم تا ارزش این استیت ها خیلی زیاد نشان داده نشود.

همانطور که در نمودارهای بالا می‌بینیم اکشن بهینه پیدا شده است و اکثر فلش ها در جهتی هستند که عامل را به هدف می‌رسانند و مسیر پیدا شده در روش های Q learning و SARSA ارزش بالاتری پیدا کرده اند.

(4

از بین روش هایی که پیاده سازی کردم :

روش Q learning سرعت همگرایی بالاتری از SARSA داشت و در حالتی که پارامتر یادگیری را تغییر می‌دادیم سرعت همگرایی SARSA افزایش پیدا می‌کرد.

هر دو الگوریتم SARSA و Q learning با اپسیلون کاهشی یادگیری بهتری داشتند و در حالتی که Q learning با اپسیلون ثابت داشتیم نوساناتی در پاسخ مشاهده کردیم.

با اپسیلون کاهشی میزان حسرت هر دو الگوریتم Q learning و SARSA کم بود ولی چون SARSA سرعت همگرایی کمتری داشت در ابتدای یادگیری حسرت بالاتری را تجربه می‌کرد.

به طور کلی روش SARSA با تنظیم خوب پارامترها بهتر از روش Q learning است چون پایداری بیشتری در طول یادگیری دارد ولی اگر از نظر سرعت همگرایی ببینیم Q learning بهتر بود.

اگر هم مدل محیط را داشتیم روش PI بهترین گزینه است.

5)

اگر این عدم قطعیت در محیط نبود و محیط deterministic بود لازم به تعداد سمپل های بالا از زندگی داخل محیط برای حل مسئله نبود و به همین دلیل سرعت همگرایی بالا می‌رفت. روش هایی مثل Q-learning در محیط های deterministic بهتر عمل می‌کنند و همانطور که بالاتر دیدیم عدم پایداری نمودارها به دلیل همین تصادفی بودن محیط است.