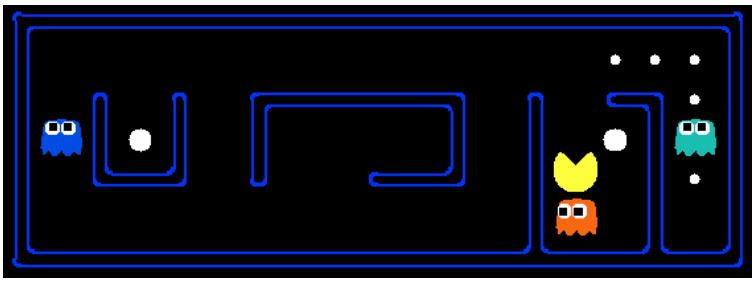
بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

یادگیری تقویتی



آیا پکمن باید نقطه را بخورد و یا فرار کند ؟

معرفی

در این پروژه شما تکرار ارزش و Q-learning را پیاده سازی خواهید کرد. ابتدا عامل خود را در دنیای شبکه ای (Gridworld) امتحان و سپس آن را روی ربات شبیه سازی شده (Crawler) و پکمن اعمال خواهید کرد.

مانند پروژه قبلی این پروژه شامل یک سیستم نمره دهی خودکار است تا نمره ی راه حل خود را روی کامپیوتر خودتان مشاهده کنید. این سیستم نمره دهی با اجرای دستور زیر روی ترمینال قابل دسترسی خواهد بود :

```
autograder.py
```

میتوانید این سیستم نمره دهی را روی یک سوال خاص هم اجرا کنید. مثلا برای نمره دهی به سوال دو خواهیم داشت :

```
autograder.py -q q2
```

همچنین برای نمره دهی به یک تست به خصوص میتوانید مانند دستور زیر عمل کنید :

```
autograder.py -t test_cases/q2/1-bridge-grid
```

کد این پروژه شامل فایل های زیر است که همگی آن ها در یک فایل فشرده برای شما آماده شده اند:

فایل هایی که باید ویرایش کنید :	
valueIterationAgents.py	عامل تکرار ارزش برای حل MDP های شناخته شده
qlearningAgents.py	عامل Q-learning برای Crawler , Gridworld و Pacman
analysis.py	فایلی که پاسخ خود به سوالات پرسیده شده در این پروژه را در آن قرار دهید
فایل هایی که باید بخوانید اما ویرایش نکنید :	
mdp.py	متود ها را روی MDP های کلی تعریف میکند
learningAgents.py	کلاس های پایه ی ValueEstimationAgent و QLearningAgent که عامل شما آن ها را گسترش خواهید داد تعریف میکند
util.py	خدماتی از قبیل util.Counter که به صورت عمده در Q-learner ها استفاده خواهد شد را ارائه میکند

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

gridworld.py	پیاده سازی Gridworld
featureExtractors.py	کلاسی برای استخراج امکانات از جفت های (state,action) که برای عامل های تقریبی Q-learning در فایل qlearningAgents.py استفاده میشود
فایل هایی که میتوانید از خواندن آن ها چشم پوشی کنید :	
environment.py	یک کلاس مجرد برای محیط های یادگیری تقویتی کلی که در فایل gridworld.py استفاده شده است
graphicsGridworldDisplay.py	نمایش گرافیکی Gridworld
graphicsUtils.py	خدمات گرافیکی
textGridworldDisplay.py	پلاگینی برای رابط متنی Gridworld
crawler.py	کد crawler و مهار تست. شما این را اجرا خواهید کرد اما ویرایشش نمیکنید
graphicsCrawlerDisplay.py	واسط گرافیکی برای ربات های crawler
autograder.py	سیستم نمره دهی خودکار پروژه
testParser.py	تجزیه آزمون نمره دهی خودکار و فایل های راه حل
testClasses.py	آزمون نمره دهی عمومی کلاس ها
test_cases/	مسیری که شامل موارد آزمون برای هر سوال است
reinforcementTestClasses.py	آزمون نمره دهی کلاس های مخصوص پروژه سوم

فایل هایی که باید ویرایش و ارسال کنید : شما بخش هایی از valueIterationAgents.py و qlearningAgents.py و analysis.py را که مشخص شده است در طول این تمرین پر خواهید کرد. لطفا سایر قسمت ها را دست نخورده باقی بگذارید. پس از تکمیل این سه فایل آن ها را زیپ کرده و نام فایل فشرده خود را هم نام با شماره دانشجویی خود قرار دهید و آن را ارسال کنید.

ارزیابی : کد شما برای ارزیابی صحت فنی توسط سیستم نمره دهی خودکار بررسی خواهد شد .لطفا نام هیچ کدام از توابع و یا کلاس های داخل کد ها را تغییر ندهید .در غیر این صورت سیستم نمره دهی خودکار کد شما را رد خواهد کرد . طبق نمرات حاصل شده از سیستم نمره دهی خودکار نمره ی سوال ۴ شما به عنوان نمره ی سوال ۶ درج شده و سایر سوال های این سیستم برای شما حذف شده است. با این حال نمره ی حاصله از سیستم نمره دهی خودکار ، نمره نهایی شما نخواهد بود و صحت پیاده سازی کد ها پایه و اساس نمره دهیست.

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

خوش آمدید

MDP ها

برای شروع Gridworld را در حالت کنترل دستی اجرا کنید (که در آن باید از کلید های جهت برای جا به جایی استفاده کنید) :

```
gridworld.py -m
```

شما از این کلاس طرح دو خروجی را خواهید دید. نقطه آبی در واقع همان عامل شماست. نکته اینکه وقتی شما دکمه بالا را میفشارید عمال تنها در ۸۰ درصد مواقع واقعا به سمت بالا حرکت میکند. زندگی عامل یک Gridworld چنین است.

شما میتوانید جنبه های زیادی از شبیه سازی را کنترل کنید. لیست کاملی از این گزینه ها برای شما فراهم شده که با اجرای دستور زیر قابل مشاهده است :

```
gridworld.py –h
```

عامل پیشفرض به صورت تصادفی حرکت میکند :

```
gridworld.py -g MazeGrid
```

با اجرای دستور بالا باید ببینید که عامل تصادفی در شبکه بدون هدف به این طرف و آن طرف حرکت میکند تا وقتی که روی خانه خروجی قرار بگیرد. مشخصا زمان طی شده تا رسیدن به خانه خروج برای یک عامل هوش مصنوعی بهترین زمان ممکن نخواهد بود.

نکته : MDP های Gridworld به این صورت است که شما اول باید وارد یک حالت pre-terminal (مربع های دو تایی در واسط گرافیکی) شوید و سپس برای خروج از بازی یک حرکت دیگر به عنوان حرکت خروج انجام دهید. (در محیط ترمینال این حالت TERMINAL_STATE خوانده میشود که در رابط گرافیکی نشان داده نمیشود)

اگر یک اپیزود را به صورت دستی اجرا کنید ، احتمالا به دلیل نرخ تخفیف (discount rate) مقدار بازگشتی کل شما از آنچیزی که انتظار داشتید کمتر خواهد شد (برای تغییر نرخ تخفیف میتوانید از d– استفاده کنید که پیشفرض آن روی ۰,۹ قرار داده شده است)

در کنار خروجی گرافیکی به خروجی ترمینال هم نگاه کنید (و یا از t– استفاده کنید تا تماما خروجی متنی مشاهده شود).

در اینجا هم مانند بازی پکمن موقعیت ها با مختصات دکارتی `(x,y)` بیان شده و تمامی آرایه ها با `[x][y]` ایندکس گذاری شده اند ، `'north'` به معنی حرکت در جهت افزایش `y` ها خواهد بود و ... . به صورت پیشفرض اکثر انتقال ها پاداشی برابر با صفر دریافت میکنند اگرچه شما میتوانید با گزینه پاداش زندگی یا همان living reward (-r) آن را تغییر دهید

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

سوال یک : تکرار ارزش

یک عامل تکرار ارزش در ValueIterationAgent بنویسید که تقریبا در فایل valueIterationAgents.py برای شما مشخص شده است. عامل تکرار ارزش شما یک برنامه ریز آفلاین است نه یک عامل یادگیری تقویتی ، بنابراین گزینه آموزش مناسب تعداد دفعات تکرار ارزشی است که (گزینه i-) باید در مرحله برنامه ریزی اولیه اش اجرا کند. ValueIterationAgent یک MDP در سازنده خود میگیرد و runValueIteration را صدا میزند که تکرار ارزش را برای تکرار های self.iterations قبل از بازگرداندن سازنده اجرا میکند.

تکرار ارزش V_k یا همان تخمین مرحله k ام مقادیر بهینه را محاسبه میکند. علاوه بر اجرای تکرار ارزش ، با استفاده از V_k متود های زیر را نیز برای ValueIterationAgent پیاده سازی کنید :

computeActionFromValues(state) : بر حسب تابع ارزش داده شده در self.values بهترین عمل را محاسبه میکند.

computeQValueFromValues(state, action) : مقدار Q-value جفت های (state, action) را که توسط تابع ارزش موجود در self.values داده شده بر میگرداند.

همه ی این مقادیر در واسط گرافیکی نشان داده شده است. اعداد داخل مربع ها نشان دهنده ارش هاست. Q-value ها در ربع مربع هاست و سیاست ها فلش هایی است که از هر مربع به بیرون رسم شده.

نکته : از نسخه دسته ای "batch" تکرار ارزش که در آن هر بردار V_k از یک بردار ثابت V_{k-1} محاسبه میشود (همانند چیزی که در لکچر خواندیم) استفاده کنید ، نه نسخه آنلاین که در آن یک بردار وزن تنها در مکان به روز رسانی میشود. این به این معنی است که وقتی ارزش یک حالت در تکرار k ام بر اساس ارزش های حالت های فرزندش به روز میشود ، این ارزش های حالت فرزند به کار رفته در محاسبات بروزرسانی ارزش باید آن هایی باشند که از تکرار k-1 ام مانده اند (حتی اگر برخی از حالت های فرزند از قبل در تکرار k ام به روز شده باشند)

نکته : سیاست کسب شده از ارزش ها در عمق k ام (که k پاداش بعدی را نشان میدهد) در واقع k+1 پاداش بعدی را نشان میدهد. (به این معنی که شما π_{k+1} را بر میگردانید) به طور مشابه Q-value ها نیز یک پاداش بیشتر از ارزش ها را نشان میدهد. (به این معنی که شما Q_{k+1} را بر میگردانید)

راهنمایی : از کلاس util.Counter موجود در فایل util.py استفاده کنید که یک دیکشنری با مقدار پیشفرض صفر است. متود هایی مانند totalCount باید کد شما را ساده تر سازند. با این حال حواستان به argMax باشد : argMax واقعی که شما میخواهید ممکن است کلیدی باشد که در counter نیست.

نکته : مطمئن شوید که شرایطی که در آن یک حالت موجود در MDP قادر به هیچ عملی نیست را کنترل میکنید.

برای تست پیاده سازی خود سیستم نمره دهی خودکار را اجرا کنید :

```
autograder.py -q q1
```

بسم الله الرحمن الرحیم

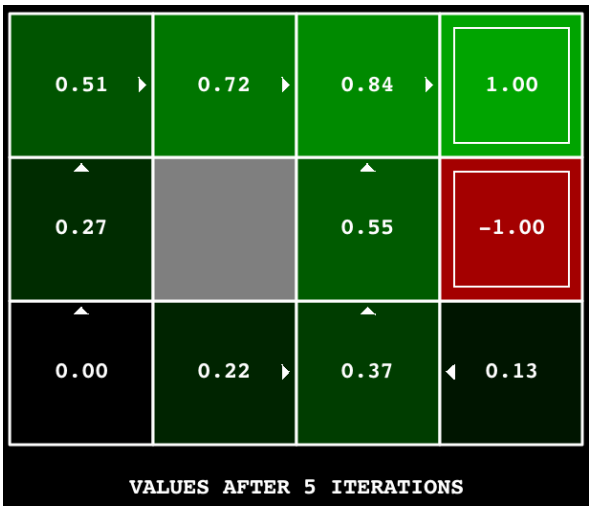
تمرین برنامه نویسی سری سوم هوش مصنوعی

دستور پیش رو ValueIterationAgent شما را بارگذاری میکند که باعث محاسبه یک سیاست و اجرای ده بار آن میشود. کلیدی را برای جا به جا شدن بین ارزش ها ، Q-value و شبیه سازی فشار دهید.
باید مشاهده کنید که ارزش در خانه آغازین V(start) که شما میتوانید از رابط گرافیکی آن را ببینید و میانگین پاداش حاصل شده ی تجربی (که بعد از ۱۰ بار اجرا چاپ میشود) به همدیگر بسیار نزدیک اند :

```
gridworld.py -a value -i 100 -k 10
```

راهنمایی : روی یک BookGrid پیشفرض اجرای تکرار ارزش برای ۵ تکرار با دستور پیش رو باید خروجی زیر برا برای شما تولید کند :

```
gridworld.py -a value -i 5
```



نمره دهی : عامل تکرار ارزش شما در یک شبکه جدید امتحان و نمره دهی خواهد شد. ما مقادیر value و Q-value و سیاست های به دست آمده شما را بعد از تعداد ثابتی تکرار بررسی خواهیم کرد

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

سوال ۲ : آنالیز عبور از پل

BridgeGrid نقشه یک دنیای شبکه ای است با یک حالت ترمینال کم پاداش و یک حالت ترمینال پر پاداش که توسط یک پل باریک جدا شده است و در هر دو طرف گذرگاه پاداش های بالای منفی وجود دارد. عامل در نزدیکی حالت کم پاداش شروع میکند. با مقدار پیشفرض نرخ تخفیف (discount) روی ۰,۹ و نویز روی ۰,۲ سیاست بهینه از پل عبور نخواهد کرد. تنها یکی از پارامتر های تخفیف و نویز را تغییر دهید تا سیاست بهینه عامل را وادار به رد شدن از پل کند. پاسخ خود را در question2() موجود در فایل analysis.py قرا دهید.(نویز به این بر میگردد که هر چند وقت یک بار یک عامل با اجرای یک عمل سر از یک حالت فرزند یا successor ناخواسته در میآورد) پاسخ پیشفرض به کد پیش رو تصویر زیر خواهد بود :

```
gridworld.py -a value -i 100 -g BridgeGrid --discount 0.9 --noise 0.2
```

نمره دهی : ما بررسی میکنیم که شما فقط یکی از پارامتر های داده شده را تغییر داده باشید و با آن تغییر یک عامل تکرار ارزش صحیح باید بتوانید از پل عبور کند. برای مشاهده نمره خود در این سوال میتوانید کد زیر را اجرا کنید :

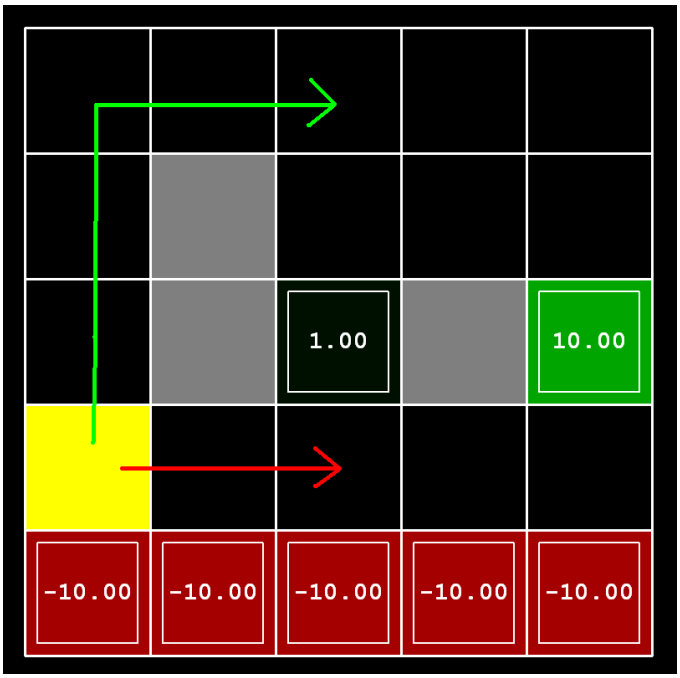
```
autograder.py -q q2
```

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

سوال ۳ : سیاست ها

طرح DiscountGrid را که در زیر نشان داده شده در نظر بگیرید. این شبکه دو حالت ترمینال با بازپرداخت مثبت دارد. (در ردیف وسطی) یک خروجی نزدیک با بازپرداخت +۱ و و یک خروجی دور با بازپرداخت +۱۰. ردیف پایین شبکه شامل حالت های ترمینال با بازپرداخت منفی هستند (که به رنگ قرمز نشان داده شده اند). حالت شروع مربع زرد است. ما باید از بین دو نوع مسیر یکی را انتخاب کنیم : (۱) مسیرهایی که خطر افتادن در ردیف قرمز را دارد و نزدیک آن حرکت میکند ؛ این مسیر ها کوتاه ترند اما خطر به دست آوردن بازپرداخت منفی بسیاری را دارند. این مسیر ها با فلش قرمز روی شکل نشان داده شده اند (۲) مسیر هایی که از ردیف قرمز دورند و از طریق لبه ی بالایی شبکه حرکت میکنند. این مسیر ها طولانی ترند و احتمال اینکه از طریق آنها بازپرداخت منفی کسب کنیم کمتر است. این مسیر ها با فلش سبز روی شکل نشان داده شده اند.



در این سوال ، شما باید تنظیماتی برای پارامتر های نرخ تخفیف (discounting) ، نویز و پاداش های زندگی (living reward) برای این MDP انتخاب کنید تا سیاست های بهینه ای از چندین نوع مختلف تولید کند. تنظیمات پارامتر های شما برای هر بخش باید این ویژگی را داشته باشد که اگر عامل شما بدون تاثیر گرفتن از هرگونه نویزی از سیاست بهینه اش پیروی کرد ، باید رفتار داده شده را از خود نشان دهد. اگر یک رفتار به خصوص برای هر تنظیماتی روی پارامتر ها حاصل نشد ، با برگرداندن مقدار رشته ای 'NOT POSSIBLE' ادعا کنید که این سیاست ممکن نیست

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

در زیر انواع سیاست های بهینه را که باید برای تولیدشان تلاش کنید مببینید :

۱- خروجی نزدیک تر (+۱) را ترجیح بده و خطر افتادن به ردیف پایینی را به جان بخر (-۱۰)

۲- خروجی نزدیک تر (+۱) را ترجیح بده و از ردیف پایینی دوری کن (-۱۰)

۳- خروجی دور تر (+۱۰) را ترجیح بده و خطر افتادن به ردیف پایینی را به جان بخر (-۱۰)

۴- خروجی دور تر (+۱) را ترجیح بده و از ردیف پایینی دوری کن (-۱۰)

۵- از هر دو خروجی و همچنین از ردیف پایینی دوری کن (بنابراین اپیزود هیچ گاه خاتمه نمیابد)

برای بررسی پاسخ خود سیستم نمره دهی خودکار را اجرا کنید :

```
autograder.py -q q3
```

question3a() تا question3e() موجود در فایل analysis.py هرکدام باید یک آیتم سه تایی از (discount, noise, living reward) برگردانند.

نکته : شما میتوانید سیاست های خود را در واسط گرافیکی چک کنید. برای مثال با با استفاده از پاسخ صحیح به 3(a) فلش موجود در خانه (0,1) باید به شرق اشاره کند ، فلش موجود در خانه (1,1) باید به شرق اشاره کند و فلش موجود در خانه (2,1) باید به شمال اشاره کند.

نکته : دربرخی از کامپیوتر ها ممکن است فلشی نبینید. در این صورت دکمه ای از کیبوردتان را برای جا به جا شدن به حالت نمایش qValue بفشارید ، و به صورت ذهنی سیاست اخذ بیشترین مقدار را در هر حالت محاسبه کنید.

نمره دهی : ما بررسی میکنیم که در هر مورد سیاست مطلوب حاصل شود

سوال ۴ : Q-Learning

باید توجه داشته باشید که عامل تکرار ارزش شما واقعا از تجربیاتش چیزی یاد نمیگیرد. بلکه آن مدل های MDP اش را برای رسیدن به یک سیاست کامل قبل از اینکه حتی با یک محیط واقعی تعامل داشته باشد میسنجد. وقتی این عامل با محیطی تعامل پیدا میکند ، به سادگی از سیاست هایی که از قبل محاسبه کرده پیروی میکند. این تمایز ممکن است در محیط های شبیه سازی شده ای مثل Gridword چندان محسوس نباشد ، اما در جهان واقعی – جایی که در آن mdp های واقعی وجود ندارند - بسیار مهم است.

اکنون شما باید یک عامل Q-learning بنویسید که ساخت و ساز آن بسیار کوچک است اما در عوض از طریق آزمون و خطای به دست آمده از فعل و انفعالاتی که با تابع update(state, action, nextState, reward) با محیط انجام میدهد چیز هایی یاد میگیرد. بدنه اصلی یک Q-learner در QLearningAgent موجود در فایل qlearningAgents.py پیاده سازی شده و شما میتوانید آن را با گزینه '-a q' انتخاب کنید.

برای این سوال شما باید متود های update و computeValueFromQValues و getQValue و computeActionFromQValues را پیاده سازی کنید.

نکته : برای computeActionFromQValues شما باید برای گرفتن رفتاری بهتر روابط را به صورت تصادفی بشکنید. تابع random.choice() شما را برای این کار کمک خواهد کرد. در یک حالت خاص عملیات هایی که عامل شما تا به حال آن ها را ندیده هنوز مقدار Q-value دارند ، مخصوصا مقداری برابر با صفر ، و در صورتی که تمام عملیات هایی که عامل شما تا آنجا دیده مقدار Q-value منفی داشته باشند ، یک عمل دیده نشده ممکن است بهینه باشد.

نکته مهم : مطمئن شوید که در کدهای مربوط به توابع computeValueFromQValues و computeActionFromQValues شما مقدار Q-value را حتما با استفاده از فراخوانی getQValue به دست آورده اید.

با به روز رسانی Q-learning میتوانید ببینید که عامل Q-learner شما تحت کنترل دستی با دستور پیش رو یاد میگیرد:

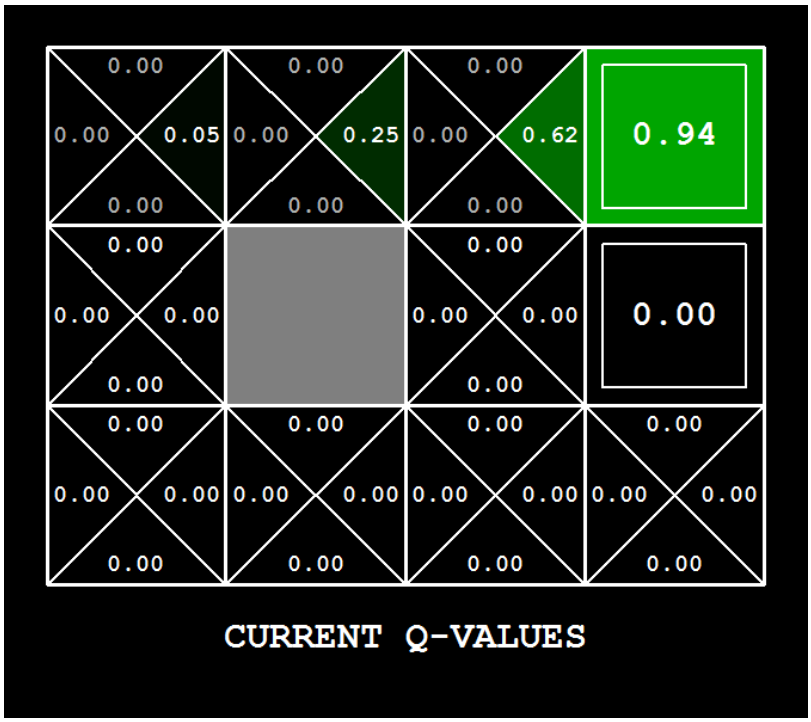
```
gridworld.py -a q -k 5 -m
```

به یاد آورید که k– تعداد اپیزود هایی را که عامل شما برای یادگیری طی میکند کنترل میکند. تماشا کنید که چطور عامل درباره حالتی که در آن بوده یاد میگیرد و نه آن حالتی که به آن میرود.

بسم الله الرحمن الرحیم

تمرین برنامه نویسی سری سوم هوش مصنوعی

راهنمایی : برای کمک در دیباگ کردن میتوانید نویز را با استفاده از گیزنه noise 0.0-- پاک کنید (البته که اینکار از جذابیت Q-learning میکاهد). اگر شما به صورت دستی عامل را به سمت شمال و سپس به سمت شرق در مسیر بهینه هدایت کنید ، با گذشت چهار اپیزود باید خروجی زیر را مشاهده کنید :



نمره دهی : ما عامل Q-learning شما را اجرا بررسی میکنیم که همان مقادیر Q-value و سیاست های پیاده سازی ما را به دست آورد. برای اینکه نمره خود را در این بخش ببینید میتوانید دستور زیر را اجرا کنید :

```
autograder.py -q q6
```

* تمرینها باید از طریق ایمیل تحویل داده شود .موضوع ایمیل ارسالی حتما به فرم مثال زیر باشد در غیر اینصورت ۲۰ درصد نمره تمرین کسر خواهد شد:

<StudentNo>-<FisrtName>-<LastName>-HW<No>-Programming-Kharazmi-AI-Fall96

83405307-Hadi-Asheri-HW1-Kharazmi-AI-Fall96

* تمرینها به آدرس ai.kharazmi.fall96@gmail.com ارسال شود.

* مهلت ارسال : ۹/۱۲ – ساعت ۱۸