دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

گزارش تمرین شماره ۳

درس یادگیری تعاملی

پاییز ۱۴۰۱

محمد سعادتی

۸۱۰۱۹۸۴۱۰

فهرست

چکیده

هدف از این تمرین آشنایی با مسائل MDP، مدلسازی و حل آنهاست. در این تمرین دو الگوریتم مهم value iteration و policy iteration را با شرایط مختلفی پیاده سازی و تحلیل می کنیم.

مسئله ۱ – آشنایی با MDP

برای هر یک از مسائل زیر یک مدل MDP ارائه می دهیم. برای این منظور جهت توصیف مدل استیت ها، اکشن ها، پاداش و انتقال بین استیت ها را تعیین کنیم.

سوال ۱

ابتدا فرضیات زیر را با هدف یکسان سازی نگاه ها به مسئله و همچنین ساده سازی مسئله و محدود کردن فضای حالات، بر روی مسئله لحاظ می کنیم:

- فرض می کنیم هدف این اپلیکیشن، کاهش درصد توده چربی در بدن انسان می باشد.
- در ارائه برنامه ورزشی، فرض می کنیم هر برنامه از ۸ حرکت تمرینی تشکیل شده است که این ۸ حرکت از یک مجموعه ۲۰ عضوی از حرکات موجود انتخاب شده اند.
- هر حرکت تمرینی شامل یک متن از توصیف نحوه انجام حرکت و مقدار کالری چربی سوزی تخمینی از انجام این حرکت با توجه به اطلاعات موجود از قبل می باشد.
- در ارائه رژیم غذایی، فرض می کنیم که یک رژیم غذایی از سه وعده شامل صبحانه، ناهار، شام تشکیل شده است به برای هر کدام به ترتیب ۳، ۵ و ۵ حالت برای انتخاب وجود دارد.
- هر از ۱۳ غذای موجود (۳تا صبحانه، ۵ تا ناهار، ۵ تا شام) شامل یک عدد که نشان دهنده میزان کالری آن غذا است می باشد.
- فرض می کنیم برنامه ورزشی و رژیم غذایی بصورت هفتگی ارزیابی، تغییر و به روز رسانی می شوند.
- انتخاب حرکات یک برنامه تمرینی یا غذای های یک رژیم غذایی بصورت رندم از حالت های ممکن صورت می گیرد؛ طوری که هدف مورد نظر از آن برنامه جدید (بطور مثال افزایش میزان کالری سوزی برنامه تمرینی یا کاهش میزان کالری رژیم غذایی) برآورده شود.
- فرض می کنیم اینکه کاهش توده چربی از چه طریقی انجام شده است (برنامه تمرینی موثر یا رژیم غذایی مناسب)، اهمیتی ندارد و صرفا هدف استفاده از حداقل یکی از این دو روش جهت کاهش توده چربی می باشد.
- طول زمان انجام رویداد ها و اتفاقات یک هفته را صفر در نظر می گیریم.

حال به مدل سازی مسئله می پردازیم:

- فضای state های مسئله (S): هر استیت را بصورت زیر تعریف می کنیم:

State = [Event, Exercise, Diet, Fitness, History]

حال اجزای این تعریف را معرفی می کنیم:

۱. رویداد "Event" : می تواند یکی از انواع زیر را داشته باشد:

- کاهش درصد توده چربی
- افزایش درصد توده چربی
- اتمام هفته

۲. رویداد "Exersice" : نشان دهنده برنامه تمرینی یک هفته که شامل ۸ حرکت تمرینی به علاوه میزان کالری لازم برای انجام این ۸ حرکت تمرینی می باشد.

۳. رویداد "Diet" : نشان دهنده رژیم غذایی یک هفته که شامل سه نوع غذا برای وعده صبحانه، ناهار، شام به علاوه مجموع میزان کالری این سه غذا می باشد.

۴. رویداد "Fitness" : نشان دهنده میزان تناسب اندام (درصد توده چربی در بدن) می باشد.

۵. رویداد "History" شامل گزارش عملکرد هر برنامه تمرینی و هر رژیم غذایی در هر یک از هفته های اخیر می باشد. با این کار می توان میزان اقبال عمومی و عملکرد مثبت یا منفی هر برنامه تمرینی یا غذایی را ارزیابی کرد تا در انتخاب یک برنامه جدید، برنامه ای که سابقه تاثیر ضعیف تری دارد با احتمال کمتری انتخاب شود.

این تعریف state مسئله را به یک مسئله Multi-State تبدیل می کند.

- فضای action ها (A) : این فضا را بصورت زیر تعریف می کنیم:

۱. اگر در استیت ای باشیم که event آن از نوع کاهش درصد توده چربی است، تنها یک عمل ممکن است و آن "حفظ برنامه تمرینی و رژیم غذایی هفته فعلی برای هفته آینده" است.

۲. اگر در استیت ای باشیم که event آن از نوع افزایش درصد توده چربی است، اعمال ممکن دو عمل "تغییر برنامه تمرینی با هدف افزایش میزان کالری حاصل از انجام برنامه تمرینی جدید" و "تغییر رژیم غذایی با هدف کاهش میزان کالری حاصل از مصرف رژیم غذایی جدید" می باشد.

۳. اگر در استیت ای باشیم که event آن از نوع اتمام هفته باشد، تنها یک عمل ممکن است و آن "اندازه گیری درصد توده چربی در بدن و به روزرسانی مقدار Fitness" می باشد.

- نحوه انتقال بین state ها ($P_{ss'}^{a}$) بصورت زیر است:

۱. در صورتی که event مربوط به state کنونی "اتمام هفته" باشد، با عمل "اندازه گیری درصد توده چربی در بدن و به روزرسانی مقدار Fitness" با توجه به میزان کاهش یا افزایش درصد توده چربی نسبت به هفته قبل، به ترتیب مقدار event به "کاهش توده چربی" و "افزایش درصد توده چربی" تغییر پیدا می کند.

۲. در صورتی که event مربوط به state کنونی "کاهش درصد توده چربی" باشد، با انجام عمل "حفظ برنامه تمرینی و رژیم غذایی هفته فعلی برای هفته آینده" ، مقدار event به "اتمام هفته" تغییر می کند.

۳. در صورتی که event مربوط به state کنونی "افزایش درصد توده چربی" باشد، با یک احتمال راندم(علت راندم انتخاب کردن جلوگیری از پیچیده شدن مسئله است) یکی از دو عمل "تغییر برنامه تمرینی با هدف افزایش میزان کالری حاصل از انجام برنامه تمرینی جدید" یا "تغییر رژیم غذایی با هدف کاهش میزان کالری حاصل از مصرف رژیم غذایی جدید" انتخاب و انجام می شود؛ که برای انتخاب برنامه جدید برای هر یک از این دو عمل، از اطلاعاتی که در تاریخچه برای برنامه های مختلف این دو عمل داریم کمک گرفته می شود تا برنامه ای که قبلا تاثیر مناسبی نداشته است با احتمال کمتری مجدد انتخاب شود. با انجام هر دوی این اعمال مقدار event به "اتمام هفته" تغییر می کند.

- محاسبه پاداش دریافتی ($R_{ss'}^{a}$) در این مسئله ساده خواهد بود:

۱. مقدار پاداش در اتمام هر هفته، برابر تفاوت درصد توده چربی هفته فعلی نسبت به هفته قبل می باشد.

۲. پاداش دیگر اعمال در تمام وضعیت ها صفر خواهد بود.

توجه کنید که چهار بخش اصلی مسئله MDP را تعریف کردیم و حال می توانیم مسئله را به صورت:

$$\text{MDP} = \{S, A, P_{ss'}^{a}, R_{ss'}^{a}\}$$

به یک مسئله MDP مدل کرد. همچنین توجه کنید که در هر مرحله هر اطلاعاتی که برای تشخیص $P_{ss'}^{a}$ و $R_{ss'}^{a}$ نیاز است را از ذات مسئله و اطلاعات انباشته شده در state همان مرحله می توان دریافت کرد و نیازی به دانستن اطلاعات دیگری نظیر مسیر رسیدن به این state نیست (توجه کنید که History که استفاده می شود در اطلاعات state ثبت شده است). بنابراین این مدل سازی یک مسئله MDP را نشان می دهد.

زیر بخش ۱

این برنامه نسبت به یک توصیه گر معمولی چه مزایا و معایبی دارد؟

به طور سنتی، مسئله recommendation به عنوان یک مسئله classification یا prediction در نظر گرفته می شد، اما اکنون به طور گسترده توافق شده است که فرمول بندی آن به عنوان یک مسئله تصمیم

گیری sequential می تواند تعامل کاربر-سیستم را بهتر منعکس کند. بنابراین، می توان آن را به عنوان یک Markov decision process (MDP) فرموله کرد و با الگوریتم های یادگیری تقویتی (RL) حل کرد. برخلاف روش‌های توصیه سنتی، از جمله collaborative filtering و content-based filtering، RL می‌تواند تعامل متوالی و پویا سیستم کاربر را مدیریت کند و تعامل طولانی‌مدت کاربر را در نظر بگیرد. اگرچه ایده استفاده از RL برای توصیه جدید نیست و حدود دو دهه است که وجود داشته است، اما عمدتاً به دلیل مشکلات مقیاس پذیری الگوریتم های سنتی RL چندان کاربردی نبود. با این حال، از زمان معرفی یادگیری تقویتی عمیق (DRL)، روند جدیدی در این زمینه پدیدار شد، که امکان اعمال RL را در مسئله توصیه با فضاهای حالت و عمل بزرگ فراهم کرد.

سوال ۲ (امتیازی)

ابتدا فرضیات زیر را با هدف یکسان سازی نگاه ها به مسئله و همچنین ساده سازی مسئله و محدود کردن فضای حالات، بر روی مسئله لحاظ می کنیم:

- فرض می کنیم هدف یک عامل، تصمیم گیری جهت شرکت یا عدم شرکت در مراسم با توجه به اوضاع شیوع بیماری در دهکده است.
- فرض می کنیم از اولین روزی که شیوع بیماری در دهکده مشاهده شد تا روز برگزاری مراسم ۳۰ روز فاصله است (یعنی اولین بار علائم بیماری در دهکده در حدود ۳۰ روز قبل تر از روز برگزاری مراسم مشاهده شده است).
- فرض می کنیم درمانگاه یا بیمارستان دهکده می تواند تعداد افراد جدیدی که در هر روز به بیماری مبتلا می شوند را سرشماری کند.
- هر فردی که تا به حال به درمانگاه مراجعه نکرده است، فرض می کند که خودش سالم است.
- مقدار جایزه نقدی مراسم را برابر ۱۰۰۰ تومان در نظر می گیریم.
- مقدار ارزش نقدی جان انسان را برابر ۱۰۰۰۰ تومان در نظر می گیریم.
- طول زمان انجام رویداد ها و اتفاقات یک روز را برابر صفر در نظر می گیریم.

حال به مدل سازی مسئله می پردازیم:

- فضای state های مسئله (S): هر استیت را بصورت زیر تعریف می کنیم:

State = [Event, SickPeople, HealthyPeople, Probability, Day]

حال اجزای این تعریف را معرفی می کنیم:

۱. رویداد Event یکی از مقادیر زیر را می تواند داشته باشد:
   - اتمام روز
   - اتمام ماه
   - ناقل بیماری(مبتلا شده به بیماری)
۲. رویداد SickPeople: نشان دهنده تعداد افراد بیمار است.
۳. رویداد HealtyPeople: نشان دهنده تعداد افراد سالم است.
۴. رویداد Probability: برابر احتمال سرایت بیماری برای یک فرد است.
۵. رویداد Day: تعداد روز های باقی مانده تا برگزاری مراسم را نشان می دهد.

این تعریف state مسئله را به یک مسئله Multi-State تبدیل می کند.

- فضای action ها (A) : این فضا را بصورت زیر تعریف می کنیم:

۱. اگر در استیت ای باشیم که event آن "ناقل بیماری" باشد، تنها یک عمل ممکن است و آن "شرکت در مراسم" است(چراکه مرگ این فرد حتمی است و مبتلا شدن یا نشدن در مراسم فرقی برایش ندارد).

۲. اگر در استیت ای باشیم که event آن از نوع "اتمام روز" است، عمل ممکن "محاسبه احتمال سرایت بیماری" می باشد.

۳. اگر در استیت ای باشیم که event آن از نوع "اتمام ماه" باشد، اعمال ممکن "شرکت در مراسم" و "عدم شرکت در مراسم" می باشد.

- نحوه انتقال بین state ها ($P_{ss'}^{a}$) بصورت زیر است:

۱. در صورتی که event مربوط به state کنونی "اتمام روز" باشد، عمل " محاسبه احتمال سرایت بیماری " انجام می شود و به استیت جدید منتقل می شود که مقدار Probability در آن برابر مقدار جدید محاسبه شده قرار می گیرد و همچنین مقدار Day نیز در آن یکی کم شده است (یک روز به برگزاری مراسم نزدیکتر شده ایم). حال اگر مقدار Day برابر صفر شده باشد، مقدار event به "اتمام ماه" تغییر پیدا می کند. در غیر این صورت مقدار event دوباره برابر "اتمام روز" قرار می گیرد.

۲. در صورتی که event مربوط به state کنونی "اتمام ماه" باشد، اگر مقدار Probability استیت کنونی بزرگتر از ۰.۵ باشد، عمل "عدم شرکت در مراسم" انجام می شود و اگر مقدار Probability کوچکتر از ۰.۵ باشد، عمل "شرکت در مراسم" صورت می گیرد.

۳. در صورتی که event مربوط به state کنونی "ناقل بیماری" باشد، تنها یک عمل ممکن است و آن "شرکت در مراسم" می باشد. با انجام این عمل مقدار event به "اتمام ماه" تغییر پیدا می کند.

- محاسبه پاداش دریافتی ($R_{ss'}^{a}$) در این مسئله به شکل زیر محاسبه می شود:

۱. مقدار پاداش در اتمام هر ماه برابر حاصل ضرب احتمال برنده شدن جایزه در مقدار جایزه (۱۰۰۰ تومان)، منهای حاصل ضرب احتمال سرایت بیماری در مقدار ارزش نقدی جان انسان (۱۰۰۰۰ تومان) می باشد.

۲. مقدار پاداش در زمان "ناقل بیماری" برابر منفی مقدار ارزش نقدی جان انسان (۱۰۰۰۰-) می باشد.

۳. پاداش دیگر اعمال در تمام وضعیت ها صفر خواهد بود.

توجه کنید که چهار بخش اصلی مسئله MDP را تعریف کردیم و حال می توانیم مسئله را به صورت:

$$\text{MDP} = \{S, A, P^{a}_{ss'}, R^{a}_{ss'}\}$$

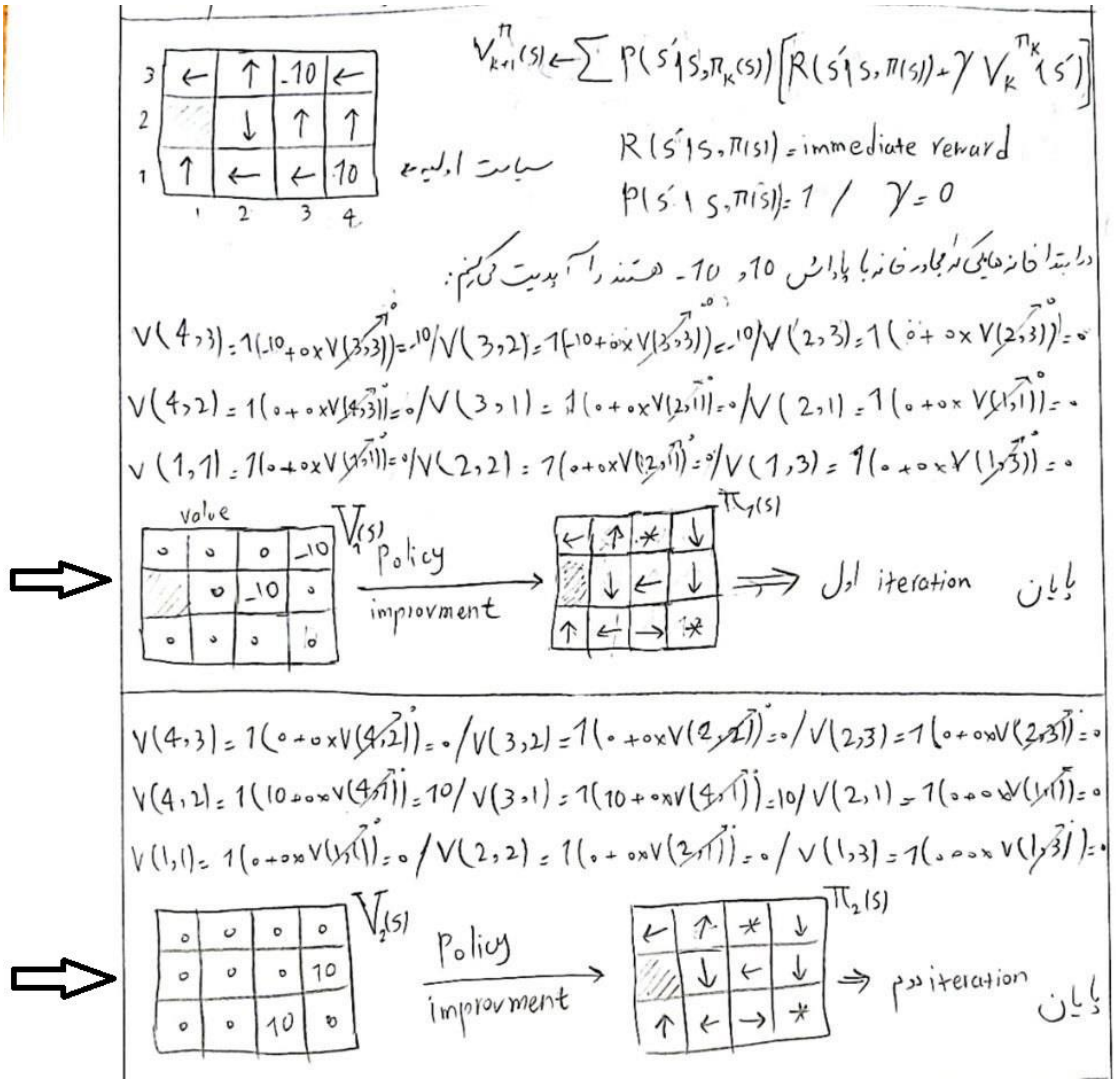
به یک مسئله MDP مدل کرد. همچنین توجه کنید که در هر مرحله هر اطلاعاتی که برای تشخیص $P^{a}_{ss'}$ و $R^{a}_{ss'}$ نیاز است را از ذات مسئله و اطلاعات انباشته شده در state همان مرحله می توان دریافت کرد و نیازی به دانستن اطلاعات دیگری نظیر مسیر رسیدن به این state نیست. بنابراین این مدل سازی یک مسئله MDP را نشان می دهد.

مسئله ۲ – پیاده سازی دستی

در این مسئله قصد داریم تا عملکرد الگوریتم های Policy Iteration و Value Iteration را به طور دستی بررسی کنیم و با جزییات این الگوریتم ها آشنا بشویم. برای این منظور، دو الگوریتم را بر روی مسئله داده به صورت گام به گام اجرا می کنیم و تغییرات در مرحله از الگوریتم را مشاهده و بررسی می کنیم.

سوال ۱

طبق الگوریتم Policy Iteration ابتدا یک سیاست راندم را در نظر می گیریم و سپس تابع هزینه را آپدیت می کنیم (همچنین مقدار ارزش اولیه تمام استیت ها برابر صفر است). این کار را ۵ مرحله تکرار می کنیم. در هر مرحله توسط معادله Bellman تابع هزینه را به روز رسانی می کنیم.



در iteration های سوم، چهارم و پنجم نیز دقیقا تمام محاسبات، مقدار ارزش استیت ها و سیاست بهینه جدید همانند iteration دوم می باشد که به دلیل تکراری بودن محاسبات در گزارش نیاورده شده است. در واقع در iteration دوم، سیاست، پایدار (stable) و همگرا می شود و دیگر بهبود نمی یابد. به همین دلیل ادامه محاسبات برای iteration های بعدی لازم نیست زیرا تمام محاسبات iteration های سوم، چهارم و پنجم مثل مثل iteration دوم خواهد بود.

مشاهده می شود که در پایان iteration اول، در مرحله policy evaluation، ارزش خانه های (۴،۳) و (۳،۲) به خاطر دریافت مجازات خانه (۳،۳) تغییر پیدا می کند. در مرحله policy improvment، سیاست ما تنها در خانه های کناری خانه Goal (خانه های (۴،۲) و (۳،۱)) و Hell (خانه های (۴،۳) و (۳،۲)) تغییر پیدا می کند که این ناشی از دریافت پاداش رفتن به خانه هدف است و سیاست سایر استیت ها بدون تغییر باقی می ماند. زیرا مقدار discount factor برابر صفر است و ارزش استیت های بعدی(استیت های کناری) در محاسبه ارزش استیت فعلی لحاظ نمی شود.

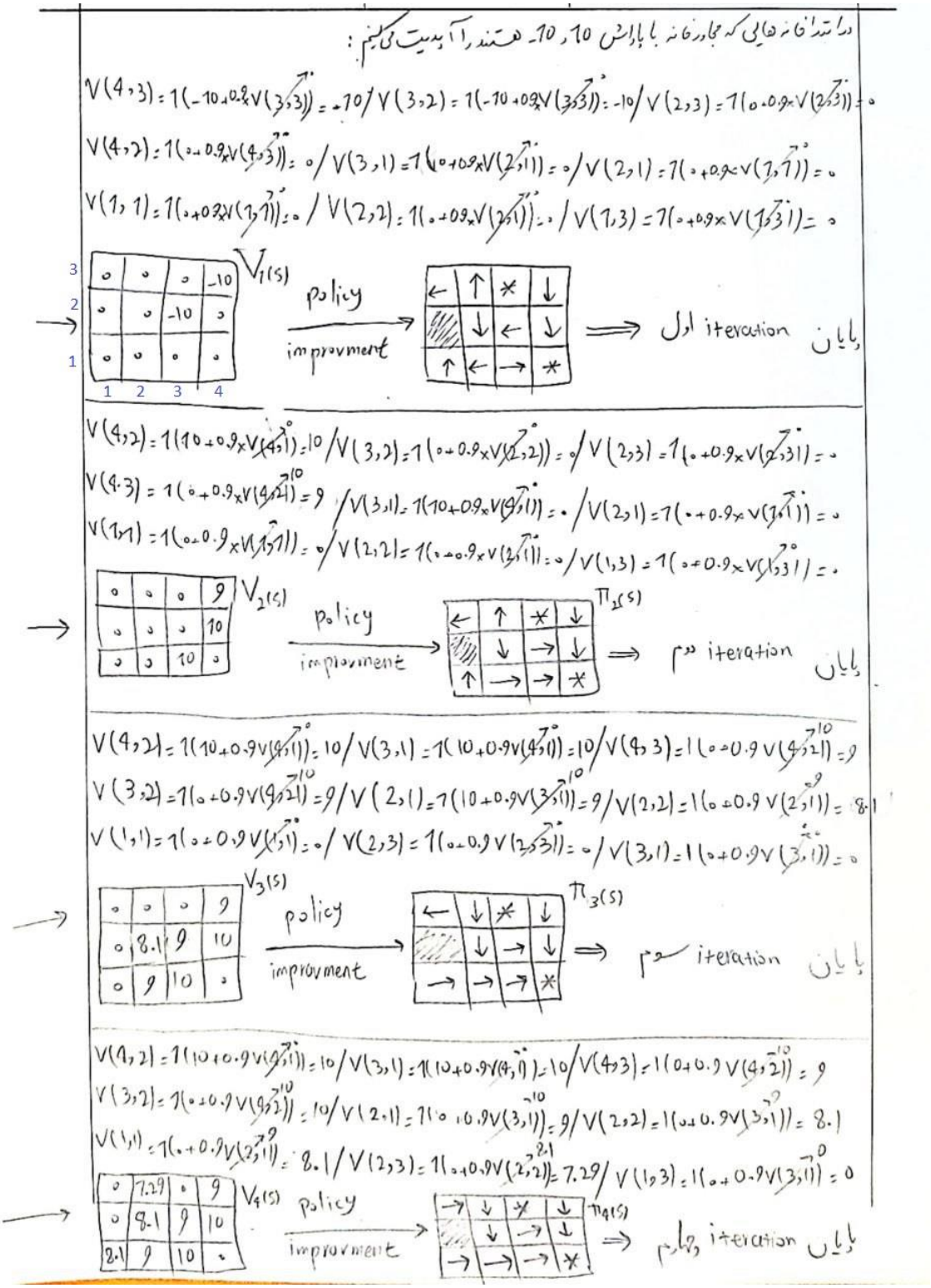
در پایان iteration دوم، در مرحله policy evaluation، ارزش خانه های (۴،۲) و (۳،۱) به خاطر دریافت پاداش خانه (۴،۱) تغییر پیدا می کند. همچنین ارزش خانه های (۴،۳) و (۳،۲) به خاطر تغییر پیدا کردن سیاست این خانه ها، برابر پاداش جا به جایی عامل که صفر است قرار می گیرد. در مرحله policy improvment، سیاست ما تغییری نمی کند و سیاست این iteration، همان سیاست iteration قبلی است. در نتیجه الگوریتم همگرا شده است و عملیات اجرا الگوریتم به پایان می رسد.

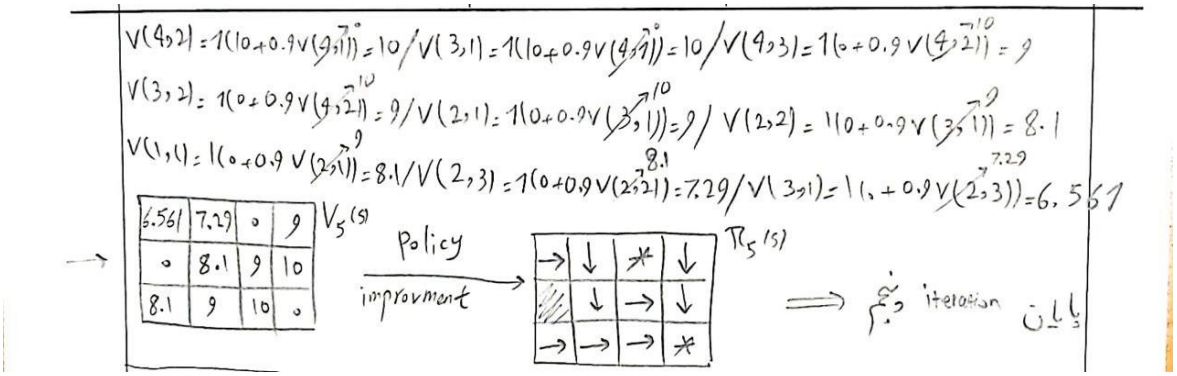
discount factor اهمیت پاداش های آینده را تعیین می کند. علت آنکه با discount factor صفر به جواب نمی رسیم آن است که ضریب ۰ عامل را تنها با در نظر گرفتن پاداش‌های فعلی کوته‌بین می‌کند، در حالی که ضریب نزدیک به ۱ باعث می‌شود که برای دریافت پاداش بلندمدت تلاش کند. اگر $\gamma=0$ باشد، عامل کاملاً نزدیک‌بین خواهد بود و فقط در مورد اعمالی که پاداش فوری ایجاد می‌کند یاد می‌گیرد. در نتیجه پاداش های بلند مدتی که توسط استیت های بعدی(استیت های کناری) تولید می شود را در ارزش یک استیت وارد نمی کند.

با توجه به اینکه discount factor صفر است، ارزش خانه های کناری هیچ وقت به ارزش یک خانه اضافه نمی شود. در نتیجه سیاست همگرا شده بعد از ۵ iteration نمی تواند ما را از هر خانه ای به خانه هدف هدایت کند. سیاست نهایی تنها در خانه های مجاور خانه هدف می تواند ما به سمت خانه هدف هدایت کند زیرا خانه های مجاور خانه هدف، به دلیل آنکه پاداش لحظه ای حاصل از رفتن به خانه هدف را دریافت کرده اند، به سیاست درستی همگرا شده اند اما سایر خانه چون چنین پاداشی را دریافت نکرده اند، دارای سیاستی می باشند که الزما ما را به خانه هدف نمی رساند.

سوال ۲

سیاست اولیه را همانند سیاست اولیه در سوال قبل در نظر می گیریم و ارزش اولیه تمام استیت ها برابر صفر است.





مشاهده می شود که بعد از ۵ iteration سیاست، پایدار (stable) و همگرا، می شود و دیگر بهبود نمی یابد

مشاهده می شود که در پایان iteration اول، در مرحله policy evaluation، ارزش خانه های (۴،۳) و (۳،۲) به خاطر دریافت مجازات خانه (۳،۳) تغییر پیدا می کند. در مرحله policy improvment، سیاست ما تنها در خانه های کناری خانه Goal (خانه های (۴،۲) و (۳،۱)) و Hell (خانه های (۴،۳) و (۳،۲)) تغییر پیدا می کند که این ناشی از دریافت پاداش رفتن به خانه هدف است و سیاست سایر استیت ها بدون تغییر باقی می ماند. زیرا ارزش استیت های بعدی(استیت های کناری) برابر صفر و در نتیجه تاثیری در ارزش سایر استیت ها ندارند.

در پایان iteration دوم، در مرحله policy evaluation، ارزش خانه های (۴،۲) و (۳،۱) به خاطر دریافت پاداش خانه (۴،۱) تغییر پیدا می کند. همچنین ارزش خانه (۳،۲) به خاطر تغییر پیدا کردن سیاست این خانه ، برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه (۲،۲) در مقدار discount factor قرار می گیرد. از طرفی ارزش خانه (۴،۳) نیز به دلیل دریافت ارزش خانه (۴،۲) ضرب در مقدار discount factor تغییر پیدا می کند. در مرحله policy improvment، سیاست ما در خانه های (۳،۲) و (۲،۱) تغییر پیدا می کند که این تغییر سیاست در این خانه ها به ترتیب ناشی از تغییر ارزش خانه (۴،۲) و (۳،۱) می باشد.

در پایان iteration سوم، در مرحله policy evaluation، ارزش خانه های (۳،۲)، (۲،۱) و (۲،۲) به خاطر تغییر پیدا کردن سیاست این خانه ها (سیاست خانه (۲،۲) تغییری نمی کند)، به ترتیب برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه های (۴،۲)، (۳،۱) و (۲،۱) در مقدار discount factor قرار می گیرد. در مرحله policy improvment، سیاست ما در خانه های (۲،۳) و (۱،۱) تغییر پیدا می کند که این تغییر سیاست در این خانه ها به ترتیب ناشی از تغییر ارزش خانه (۲،۲) و (۲،۱) می باشد.

در پایان iteration چهارم، در مرحله policy evaluation، ارزش خانه های (۲،۳) و (۱،۱) به خاطر تغییر پیدا کردن سیاست این خانه ها ، به ترتیب برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه های (۲،۲) و (۲،۱) در مقدار discount factor قرار می گیرد. در مرحله policy improvment، سیاست ما در خانه (۱،۳) تغییر پیدا می کند که این تغییر سیاست در این خانه ناشی از تغییر ارزش خانه (۲،۳) می باشد.

در پایان iteration پنجم، در مرحله policy evaluation، ارزش خانه (۱،۳) به خاطر تغییر پیدا کردن سیاست این خانه ها ، به ترتیت برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه (۲،۳) در مقدار discount factor قرار می گیرد. در مرحله policy improvment، سیاست ما نسبت به سیاست iteration قبل تغییری پیدا نمی کند. این نشان دهنده آن است که الگوریتم به یک سیاست همگرا و پایدار (stable) شده است.

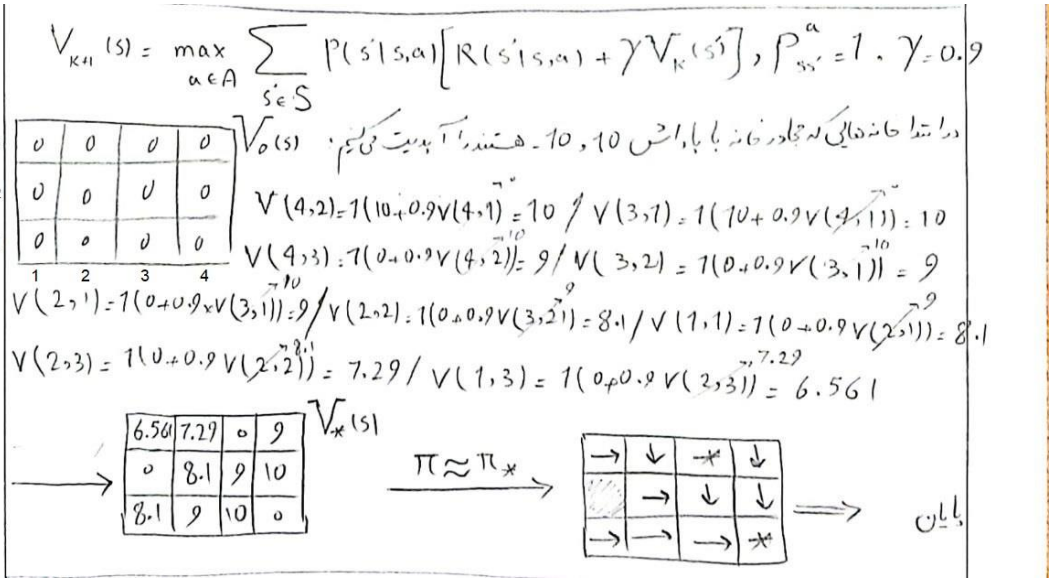
سیاست همگرا شده در پایان iteration پنجم یک سیاست بهینه است که عامل را از هر خانه ای به سمت خانه هدف هدایت می کند.

discount factor اساساً تعیین می‌کند که عوامل یادگیری تقویتی چقدر به پاداش‌ها در آینده دور نسبت به پاداش‌های آینده نزدیک اهمیت می‌دهند. ضریب ۰ عامل را تنها با در نظر گرفتن پاداش‌های فعلی کوته‌بین می‌کند، در حالی که ضریب نزدیک به ۱ باعث می‌شود که برای دریافت پاداش بلندمدت تلاش کند. اگر $\gamma=1$ باشد، عامل هر یک از اقدامات خود را بر اساس مجموع کل پاداش های آینده خود ارزیابی می کند. اگر discount factor از ۱ بیشتر شود، مقادیر عمل ممکن است متفاوت باشد.

تغییر مقدار discount factor در این سوال به یک مقدار غیر صفر(بر خلاف سوال قبل که برابر صفر بود)، باعث می شود تا در هنگام محاسبه ارزش یک استیت با استفاده از معادله Bellman، ارزش استیت های بعدی(خانه های کناری) نیز در ارزش یک استیت لحاظ شود. این باعث می شود تا به تدریج هر چه از خانه هدف دورتر می شویم، ارزش استیت ها کم بشود و برابر حداقل مقدار ممکن نیز نباشند. در نتیجه یک استیت می تواند با استفاده از ارزش استیت ای که در مجموعه استیت های قابل انتقال از استیت فعلی، دارای بیشترین ارزش است و همچنین برابر مقادیر حداقلی نیز نیست، بهترین اکشن و سیاست را انتخاب کند طوری که با پیروی از آن سیاست به خانه هدف نزدیکتر شود.

سوال ۳

در این سوال الگوریتم Value Iteration را بطور دستی اجرا و بررسی می کنیم. مقدار ارزش اولیه تمام استیت ها برابر صفر است. در هر مرحله توسط معادله Bellman تابع هزینه را به روز رسانی می کنیم.



جهت جلوگیری از انجام محاسبات اضافه، در هر حلقه از الگوریتم، ما محاسبه ارزش استیت ها را از خانه های نزدیک هدف شروع کردیم و سپس به خانه های دورتر برای محاسبه ارزش آن استیت رفتیم.

در ابتدا ارزش خانه های (۴،۲) و (۳،۱) به ترتیب به خاطر انجام اکشن "پایین" و "راست" و به دلیل دریافت پاداش رفتن به خانه (۴،۱) تغییر پیدا می کند. سپس ارزش خانه های (۴،۳)، (۳،۲) و (۲،۱) به ترتیب به خاطر انجام اکشن "پایین"، "پایین"، "راست" و به دلیل تغییر پیدا کردن ارزش خانه های (۴،۲) ، (۳،۱) و (۳،۱) ، به ترتیب برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه های (۴،۲)، (۳،۱) و (۳،۱) در مقدار discount factor قرار می گیرد. در مرحله بعد ارزش خانه های (۲،۲) و (۱،۱) به ترتیب به خاطر انجام اکشن "راست"، "راست" و به دلیل تغییر پیدا کردن ارزش خانه های (۳،۲) ، (۲،۱) ، به ترتیب برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه های (۳،۲)، (۲،۱) در مقدار discount factor قرار می گیرد. سپس ارزش خانه (۲،۳) به خاطر انجام اکشن "پایین" و به دلیل تغییر پیدا کردن ارزش خانه (۲،۲)، برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه های (۲،۲) در مقدار discount factor قرار می گیرد. در مرحله بعد ارزش خانه (۱،۳) به خاطر انجام اکشن "راست" و به دلیل تغییر پیدا کردن ارزش خانه (۲،۳)، برابر پاداش جا به جایی عامل که صفر است به علاوه حاصل ضرب ارزش خانه های (۲،۳) در مقدار discount factor قرار می گیرد. در این مرحله دقت الگوریتم به ترشولد مدنظر می رسد و لوپ اصلی الگوریتم پایان می یابد.

حال به پیدا کردن سیاست بهینه برای هر استیت می پردازیم. در هر استیت، اکشنی به عنوان سیاست بهینه در آن استیت انتخاب می شود که مقدار معادله Bellman از انجام آن اکشن نسبت به سایر اکشن های مجاز در آن استیت بیشتر باشد.

زیر بخش ۱

آیا مفهوم خاصی برای ارزش بدست آمده در هر مرحله وجود دارد؟

ارزش استیت ها در هر مرحله بیان می کند که برای یک استیت، چقدر حضور در آن استیت نسبت به سایر استیت بهتر یا بدتر است و ما را می تواند به استیت هدف نزدیکتر بکند. هر چه یک استیت ارزش بالاتری داشته باشد، یعنی تلاش برای رسیدن به آن برای ما سودمند تر است و آن استیت، بهتر می تواند ما برای رسیدن به استیت هدف هدایت بکند.

همچنین ارزش بدست آمده در مرحله نشان دهنده میزان دقت الگوریتم در تخمین ارزش ها می باشد. با مقایسه ارزش بدست آمده در یک مرحله با مرحله قبل، می توان دقت تخمین هایمان از ارزش محاسبه شده برای استیت ها را تعیین کرد. با استفاده از این تعیین دقت تخمین می توان فهمید که آیا الگوریتم باید ادامه پیدا کند یا مراحل الگوریتم به پایان رسیده است.

زیر بخش ۲

آیا لزوما در هر مرحله ی این الگوریتم، سیاستی متناظر با ارزش های بدست آمده وجود دارد؟

خیر. در هر مرحله از الگوریتم، با توجه به ترتیب انتخاب استیت ها برای بروزرسانی ارزش آنها، مقادیر متفاوتی برای ارزش یک استیت در پایان یک مرحله از الگوریتم می تواند ایجاد شود. در نتیجه در پایان یک مرحله از الگوریتم، ارزش یک استیت می تواند به شکل های مختلفی تغییر کند (دارای مقادیر متفاوتی باشد) که این تفاوت ناشی از ترتیب ما در انتخاب استیت ها برای محاسبه ارزش آنها می باشد. پس با توجه به مقدار ارزش استیت ها در هر مرحله از الگوریتم، ممکن برای برخی از استیت ها نتوان سیاستی تعیین کرد زیرا هنوز مقدار ارزش آن استیت و استیت های کناری اش به مقدار مناسبی محاسبه نشده است و ممکن است همچنان دارای مقادیر اولیه باشند. این باعث می شود تا نتوان برای یک استیت بهترین اکشن (سیاست) را پیدا کرد زیرا هنوز ارزش استیت های بعدی آن محاسبه نشده است. (فرض کرده ایم حالتی که نمی توان یک سیاست متناظر با ارزش های بدست آمده به طور یکتا تعیین کرد و سیاست به طور راندم تعیین می شود قابل قبول نیست.)

سوال ۴

فرض کنید به جای استفاده از مقدار اولیه ی رندم برای ارزش استیت ها، از یک ارزشی استفاده کنید که سیاست متناظر با آن سیاست بهتری نسبت به سیاست متناظر با ارزش های با مقدار رندم باشد.

زیر بخش ۱

آیا این کار ممکن است در iteration های کمتری به جواب بهینه رسید؟

بله – از آنجایی که ترتیب انتخاب استیت ها در محاسبه ارزش استیت ها تاثیر دارد، اگر سیاست ما طوری باشد استیت های بعدی یک استیت که در تعیین ارزش یک استیت تاثیر دارند، زودتر ارزششان محاسبه شود، ارزش آن استیت نیز در هنگام محاسبه مقدار بهتری می گیرد. در واقع اگر ارزش اولیه استیت ها به شکلی باشد که سیاست متناظر با آن باعث بشود ابتدا استیت هایی که ارزش بیشتری خواهند داشت و در مقدار ارزش یک استیت موثرتر هستند، زودتر محاسبه بشوند، تعداد گام های غیر موثر الگوریتم کاهش می یابد. این باعث می شود تا سیاست های ابتدایی شباهت بیشتری به سیاست بهینه داشته باشند که در نتیجه آن تعداد محاسبات بیهوده الگوریتم (محاسباتی که ارزش حاصل از آنها تغییر و بهبودی در سیاست ایجاد نمی کند) کمتر بشود. در نتیجه الگوریتم ما زودتر به یک سیاستی همگرا می شود و تعداد iteration های الگوریتم کاهش پیدا می کند.

زیر بخش ۲

آیا این عمل تاثیری روی نرخ همگرایی دارد؟

بله – این کار باعث می شود تا میزان تغییر ارزش و سیاست از یک مرحله به مرحله دیگر (نرخ همگرایی) کمتر بشود. زیرا ارزش و سیاست اولیه شباهت و همخوانی بیشتری با ارزش و سیاست بهینه دارند و به همین دلیل کمتر نیاز به تغییر و به روزرسانی دارند. در واقع در هر مرحله تغییرات کمتری در ارزش و سیاست مان مشاهده می کنیم زیرا ارزش اولیه منجر به سیاستی شده است که به ارزش و سیاست بهینه نزدیک تر است. در نتیجه این عمل، تعداد گام و iteration های الگوریتم کاهش پیدا می کند و الگوریتم نیز زودتر همگرا می شود. این وضعیت همانند حالتی است که ما از وسط گام های یک الگوریتمی که با یک ارزش و سیاست راندم کار خود را شروع کرده است، محاسبات خود را آغاز کنیم و دیگر لازم نباشد برخی گام های ابتدایی موجود در حالت ارزش دهی با مقادیر راندم را طی کنیم.

مسئله ۳ – شبیه سازی

هدف سوال

در این بخش قرار است مسئله روی دریاچه یخی را حل کنیم. برای این کار از الگوریتم های Policy Iteration و Value Iteration کمک می گیریم. با استفاده از این الگوریتم ها، ارزش استیت ها و سیاست بهینه برای رسیدن به مقصد را در شرایط مختلف پیدا و تحلیل می کنیم.

توضیح پیاده سازی

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3** قرار دارد.

تابع ***show_states_value_on_map*** مقدار ارزش استیت ها را بر روی نقشه نشان می دهد.

تابع ***show_actions_on_map*** سیاست بهینه را بر روی نقشه نشان می دهد.

در کلاس **ValueIteration** الگوریتم Value Iteration پیاده سازی شده است:

- در تابع ***value_estimation*** الگوریتم Value Iteration, for estimating $\pi \sim \pi_*$ پیاده سازی شده است.
- تابع ***get_optimal_policy*** سیاست بهینه در هر استیت را خروجی می دهد.
- تابع ***get_state_values*** مقدار ارزش استیت ها را خروجی می دهد.
- تابع ***get_q_values*** مقدار ارزش استیت-اکشن ها را خروجی می دهد.

در کلاس **PolicyIteration** الگوریتم Policy Iteration پیاده سازی شده است:

- در تابع ***policy_estimation*** الگوریتم Policy Iteration (using iterative policy evaluation) for estimating $\pi \sim \pi_*$ پیاده سازی شده است.
- در تابع ***policy_evaluation*** قسمت policy evaluation الگوریتم Policy Iteration پیاده سازی شده است.
- در تابع ***policy_improvement*** قسمت policy improvement الگوریتم Policy Iteration پیاده سازی شده است.
- تابع ***get_optimal_policy*** سیاست بهینه در هر استیت را خروجی می دهد.
- تابع ***get_state_values*** مقدار ارزش استیت ها را خروجی می دهد.
- تابع ***get_q_values*** مقدار ارزش استیت-اکشن ها را خروجی می دهد.

بخش اول

هدف سوال

در این بخش به اجرای دو الگوریتم Policy Iteration و Value Iteration بر روی مسئله دریاچه یخی با مقادیر داده شده برای پارامتر های مختلف(احتمال شکستن، مقدار پاداش، میزان لغزندگی) می پردازیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 1** قرار دارد

0.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.0001	0.0001	1.0000	1.0000	1.0000	1.0000
1.0000	0.0001	1.0000	1.0000	1.0000	1.0000
1.0000	0.0001	0.0001	0.0001	0.0001	0.0001
1.0000	1.0000	1.0000	1.0000	1.0000	0.0001
1.0000	1.0000	1.0000	1.0000	1.0000	0.0000

Figure ۱نقشه بخش اول

Value Iteration

در این قسمت الگوریتم Value Iteration را بر روی مسئله دریاچه یخی اجرا می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 1 - Value Iteration** قرار دارد.

states value :

26.1929	30.0812	17.3231	22.7638	29.6611	36.8419
30.6258	35.6471	31.9859	38.4259	46.1634	54.3519
35.9200	41.8198	48.6298	56.2760	64.9640	74.3257
41.3085	48.5310	56.3703	65.1780	74.9210	85.6277
25.6742	41.4664	48.9862	64.8993	85.4633	97.6371
21.5512	37.0133	54.6250	74.5789	97.0272	0.0000

Figure ۲ مقدار ارزش استیت ها

optimal policy :

↓	↓	↓	↓	↓	↓
→	↓	↓	↓	↓	↓
→	↓	↓	↓	↓	↓
→	→	→	→	→	↓
→	↑	↑	→	→	↓
→	→	→	→	→	*

Figure ۳ سیاست بهینه

Policy Iteration

در این قسمت الگوریتم Policy Iteration را بر روی مسئله دریاچه یخی اجرا می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 1 - Policy Iteration** قرار دارد.

```
states value :

-------------------------------------------------------------
| 26.1929 | 30.0812 | 17.3231 | 22.7638 | 29.6611 | 36.8419 |
-------------------------------------------------------------
| 30.6258 | 35.6471 | 31.9859 | 38.4259 | 46.1634 | 54.3519 |
-------------------------------------------------------------
| 35.9200 | 41.8198 | 48.6298 | 56.2760 | 64.9640 | 74.3257 |
-------------------------------------------------------------
| 41.3085 | 48.5310 | 56.3703 | 65.1780 | 74.9210 | 85.6277 |
-------------------------------------------------------------
| 25.6742 | 41.4664 | 48.9862 | 64.8993 | 85.4633 | 97.6371 |
-------------------------------------------------------------
| 21.5512 | 37.0133 | 54.6250 | 74.5789 | 97.0272 | 0.0000 |
-------------------------------------------------------------
```

Figure ۴ مقدار ارزش استیت ها

```
optimal policy :

-------------------------
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | → | → | → | → | ↓ |
-------------------------
| → | ↑ | ↑ | → | → | ↓ |
-------------------------
| → | → | → | → | → | * |
-------------------------
```

Figure ۵ سیاست بهینه

روند اجرای کد پیاده‌سازی

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل این بخش و کد های این بخش را اجرا کنید.

بخش دوم

هدف سوال

در این بخش، ابتدا تغییراتی بر روی نقشه دریاچه اعمال می کنیم. سپس به اجرای دو الگوریتم Policy Iteration و Value Iteration بر روی مسئله دریاچه یخی با مقادیر داده شده برای پارامتر های مختلف(احتمال شکستن، مقدار پاداش، میزان لغزندگی) می پردازیم. در نهایت نتایج بدست آمده در این بخش را با بخش قبلی مقایسه و تحلیل می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 2** قرار دارد.

0.0000	0.8444	0.3343	0.3058	0.8208	0.4955
0.0010	0.0010	0.7536	0.4510	0.8006	0.3135
0.3768	0.0010	0.8268	0.8268	0.8908	0.5312
0.6320	0.0010	0.0010	0.0010	0.0010	0.0010
0.9490	0.8463	0.6388	0.1015	0.2757	0.0010
0.6810	0.1309	0.2624	0.1043	0.1053	0.0000

Figure ۶ نقشه بخش دوم

Value Iteration

در این قسمت الگوریتم Value Iteration را بر روی مسئله دریاچه یخی اجرا می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 2 - Value Iteration** قرار دارد.

```
states value :

-------------------------------------------------------------
| 4.1375 | 5.4381 | 0.0744 | 5.8946 | 11.2824 | 22.1840 |
-------------------------------------------------------------
| 7.5472 | 10.7228 | 10.4150 | 15.8092 | 23.9532 | 35.2093 |
-------------------------------------------------------------
| 12.5000 | 17.8140 | 24.7531 | 33.0679 | 43.1218 | 54.3110 |
-------------------------------------------------------------
| 16.5526 | 24.9331 | 34.6425 | 45.5745 | 57.5673 | 71.6122 |
-------------------------------------------------------------
| 13.2761 | 27.7960 | 43.6927 | 57.2155 | 73.8153 | 89.9020 |
-------------------------------------------------------------
| 25.1811 | 37.5886 | 53.8614 | 70.5729 | 89.5710 | 0.0000 |
-------------------------------------------------------------
```

Figure ۷ مقدار ارزش استیت ها

```
optimal policy :

-------------------------
| ↓ | ↓ | → | ↓ | → | ↓ |
-------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | → | → | → | → | ↓ |
-------------------------
| → | → | → | → | → | ↓ |
-------------------------
| → | → | → | → | → | * |
-------------------------
```

Figure ۸ سیاست بهینه

Policy Iteration

در این قسمت الگوریتم Policy Iteration را بر روی مسئله دریاچه یخی اجرا می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 2 - Policy Iteration** قرار دارد.

```
states value :

-------------------------------------------------------------
| 4.1375 | 5.4381 | 0.0744 | 5.8946 | 11.2824 | 22.1840 |
-------------------------------------------------------------
| 7.5472 | 10.7228 | 10.4150 | 15.8092 | 23.9532 | 35.2093 |
-------------------------------------------------------------
| 12.5000 | 17.8140 | 24.7531 | 33.0679 | 43.1218 | 54.3110 |
-------------------------------------------------------------
| 16.5526 | 24.9331 | 34.6425 | 45.5745 | 57.5673 | 71.6122 |
-------------------------------------------------------------
| 13.2761 | 27.7960 | 43.6927 | 57.2155 | 73.8153 | 89.9020 |
-------------------------------------------------------------
| 25.1811 | 37.5886 | 53.8614 | 70.5729 | 89.5710 | 0.0000 |
-------------------------------------------------------------
```

Figure ۹ مقدار ارزش استیت ها

```
optimal policy :

-------------------------
| ↓ | ↓ | → | ↓ | → | ↓ |
-------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------
| → | → | → | → | → | ↓ |
-------------------------
| → | → | → | → | → | ↓ |
-------------------------
| → | → | → | → | → | * |
-------------------------
```

Figure ۱۰ سیاست بهینه

روند اجرای کد پیاده‌سازی

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل بخش اول و کد های این بخش را اجرا کنید.

مقایسه نتایج با قسمت قبل

ارزش استیت ها در این بخش نسبت به بخش قبل، تقریبا در اکثر استیت ها مقدار کمتری نسبت به بخش قبل است؛ علت آن است که عدم قطعیت بیشتر در این بخش نسبت به بخش قبل در انجام اکشن انتخاب شده(۰.۷ در این بخش و ۰.۹۴ در بخش قبل) ، باعث می شود تا پاداش و ارزش حاصل از اکشن انتخاب شده در قسمت محاسبه ارزش استیت فعلی در دو الگوریتم $\max_a (\text{Sum}_{s,r} (p(s', r \mid s, a)[r$)))

[(+ γ V(s')] (قسمت قرمز رنگ) کاهش بیابد. در واقع چون احتمال لغزندگی بالاتر رفته است، پاداش اکشن انتخاب شده برای یک استیت تاثیر کمتری (کمتر شدن احتمال transition – قسمت قرمز رنگ) (۰.۷ در این بخش و ۰.۹۴ در بخش قبل) در ارزش آن استیت می گذارد و پاداش و ارزش حاصل از انجام اکشن های انتخاب نشده در آن استیت، تاثیری بیشتری(۰.۱ در این بخش و ۰.۰۲ در بخش قبل) در ارزش یک استیت می گذارند. با توجه به اینکه ارزش استیت ها در این بخش نسبت به بخش قبل متفاوت است، به طبع آن سیاست بهینه نیز در این بخش در مقایسه با بخش قبل متفاوت است.

مقایسه سرعت همگرایی الگوریتم ها

میانگین سرعت همگرایی الگوریتم ها در دو بخش اول و دوم برای ۵۰ بار اجرای هر الگوریتم در هر بخش به شرح زیر است:

	بخش اول	بخش دوم
Value Iteration	۱.۱۱۶۱۶۸۹۵۱۹۸۸۲۲۰۳	۱.۹۶۵۳۰۶۰۰۰۷۰۹۵۳۳۸
Policy Iteration	۲۱.۵۹۵۹۴۳۲۱۲۵۰۹۱۵۵	۱۶.۴۹۶۸۴۲۸۶۱۱۷۵۵۳۷

در policy iteration، با یک policy ثابت شروع می کنیم. برعکس، در value iteration، با انتخاب تابع value شروع می کنیم. سپس، در هر دو الگوریتم، به طور مکرر بهبود می‌یابیم تا به همگرایی برسیم.

الگوریتم policy iteration، policy را به روز می کند. در عوض الگوریتم value iteration روی تابع value تکرار می شود. با این حال، هر دو الگوریتم به طور ضمنی policy و تابع state-value را در هر تکرار به روز می کنند.

هر دو الگوریتم تضمین شده است که در پایان به یک policy بهینه همگرا شوند.

در Policy Iteration ، در هر مرحله، policy evaluation تا زمان همگرایی اجرا می شود، سپس policy به روز می شود و فرآیند تکرار می شود.

در مقابل، Value Iteration تنها یک iteration واحد از policy evaluation را در هر مرحله انجام می دهد. سپس، برای هر state ، حداکثر action-value ، state-value تخمین زده شده است. هنگامی که این state-values به state-values بهینه همگرا شدند، می توان policy بهینه را بدست آورد. در عمل این بسیار بهتر از Policy Iteration عمل می کند و تابع state-value بهینه را در مراحل بسیار کمتری پیدا می کند.

بخش سوم

هدف سوال

در این قسمت می خواهیم تاثیر تابع پاداش و مقدار ترشولد θ در الگوریتم ها را بررسی کنیم. برای این منظور نقشه محیط را تغییر داده و ابعاد دریاچه را به صورت ۱۵ در ۱۵ در نظر می گیریم. همچنین شرایط بخش دوم (احتمال شکستن، مقدار پاداش، میزان لغزندگی) را بر روی این نقشه جدید اعمال می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 3** قرار دارد.

0.0000	0.4520	0.5950	0.6061	0.7932	0.7197	0.2585	0.6634	0.5232	0.0378	0.7218	0.9371	0.6105	0.7366	0.6932
0.0010	0.0010	0.9939	0.9846	0.0024	0.9573	0.6653	0.9827	0.4317	0.7456	0.0247	0.3842	0.2294	0.6634	0.4343
0.8383	0.0010	0.0010	0.0010	0.0010	0.4788	0.3056	0.9012	0.6050	0.6653	0.6583	0.2466	0.9152	0.7013	0.4871
0.1692	0.5775	0.1076	0.1591	0.0010	0.3518	0.6244	0.7026	0.3913	0.8066	0.0295	0.9734	0.8439	0.5912	0.0005
0.1440	0.6107	0.1992	0.9467	0.0010	0.4791	0.8437	0.0493	0.9428	0.5429	0.7290	0.0878	0.0083	0.9437	0.2499
0.2712	0.1510	0.8071	0.3817	0.0010	0.0010	0.2831	0.6867	0.7443	0.9140	0.8392	0.2167	0.7163	0.0985	0.6631
0.0936	0.1786	0.0865	0.9699	0.8448	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.0010	0.2267	0.5241	0.2428
0.1761	0.8059	0.9905	0.0121	0.0008	0.5723	0.9803	0.3645	0.6211	0.2615	0.9099	0.0010	0.9671	0.1006	0.1122
0.0488	0.6707	0.0755	0.8038	0.3957	0.7134	0.1600	0.2228	0.0079	0.1332	0.9666	0.0010	0.6056	0.8237	0.1465
0.7942	0.2243	0.1573	0.2286	0.4545	0.2505	0.6105	0.7919	0.2427	0.7240	0.3716	0.0010	0.1378	0.4928	0.4327
0.4227	0.2723	0.2890	0.5268	0.6244	0.8631	0.7457	0.3508	0.9613	0.1801	0.6988	0.0010	0.0829	0.5639	0.4337
0.8263	0.3513	0.6711	0.9126	0.2859	0.9794	0.1129	0.4107	0.1958	0.1167	0.6121	0.0010	0.0010	0.0010	0.0010
0.2046	0.8016	0.2783	0.9527	0.9989	0.4543	0.7732	0.6200	0.6368	0.1192	0.1372	0.9705	0.2537	0.1759	0.0010
0.0098	0.1900	0.8140	0.4966	0.7114	0.8010	0.7622	0.2487	0.3123	0.3603	0.3399	0.6617	0.7923	0.1449	0.0010
0.0460	0.0803	0.6307	0.7311	0.0880	0.4431	0.5792	0.7494	0.0239	0.3370	0.1317	0.4027	0.0683	0.5118	0.0000

Figure ۱۱ نقشه بخش سوم

الف)

زیر بخش ۱

آیا تغییر مقدار ترشولد در الگوریتم value iteration، در یافتن سیاست بهینه تاثیری دارد؟

بله. می دانیم الگوریتم value iteration در نهایت به یک سیاستی همگرا می شود؛ یعنی این الگوریتم پایان پذیر است. اگر مقدار ترشولد برابر مقدار بزرگی قرار بگیرد، ممکن است الگوریتم به سرعت همگرا شود ولی سیاست بدست آمده سیاست بهینه نباشد یعنی بهبود کافی روی سیاست های ابتدایی انجام نشود و همان سیاست های اولیه به عنوان سیاست نهایی و همگرا شده در نظر گرفته بشوند. همچنین اگر مقدار ترشولد برابر مقدار کوچکی قرار بگیرد، ممکن است الگوریتم در مدت زمان خیلی زیادی همگرا بشود. با مقدار ترشولد کوچک نسبت به مقدار بزرگ، احتمال بالاتری در پیدا کردن سیاست بهینه وجود دارد(سیاست همگرا شده سیاست بهینه باشد) اما ممکن است این کار خیلی طول بکشد و الگوریتم دیر همگرا بشود.

زیر بخش ۲

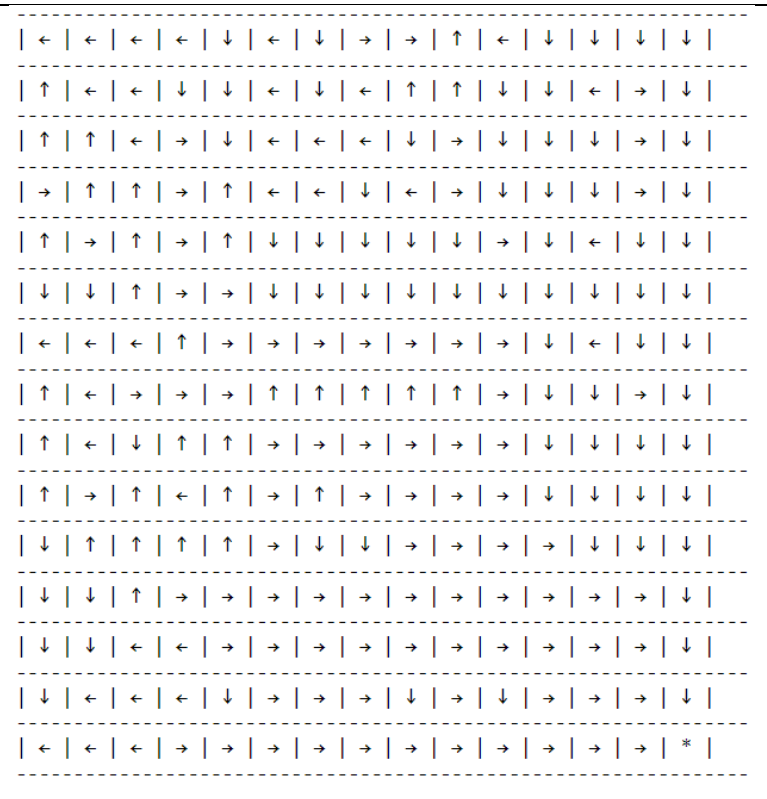
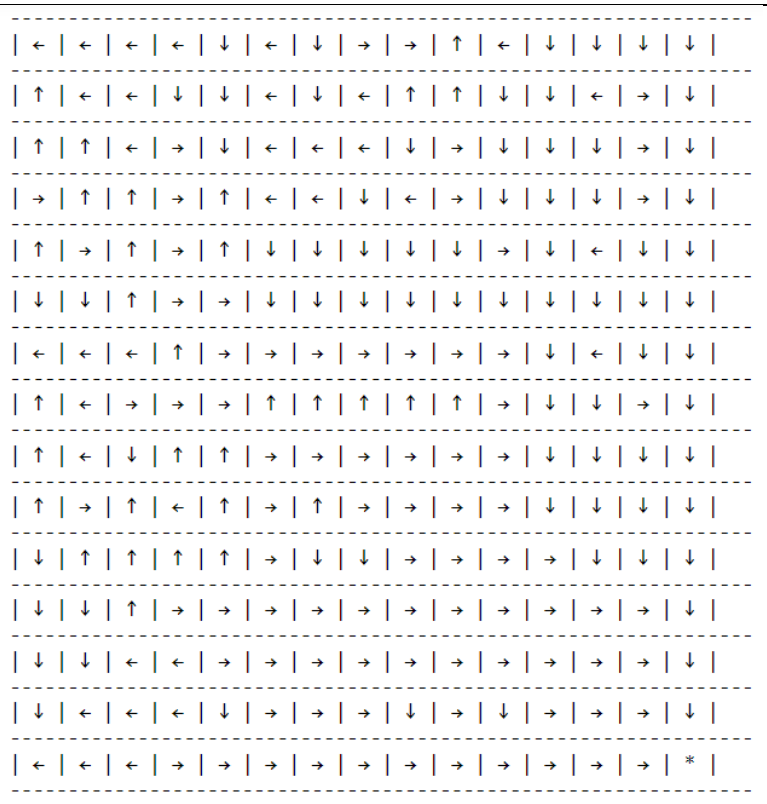
مقدار ترشولد باید چگونه باشد تا یک سیاست بهینه حاصل شود؟

برای پیدا کردن پاسخ این سوال ما الگوریتم value iteration را با مقادیر ۱، ۰.۱، ۰.۰۱، ۰.۰۰۱، ۰.۰۰۰۱، ۰.۰۰۰۰۱، ۰.۰۰۰۰۰۱ به عنوان ترشولد اجرا می کنیم و نتایج را بررسی می کنیم.

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت **Problem 3 – Part 3 - A** قرار دارد.

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل بخش اول و کد های این بخش را اجرا کنید.

Optimal policy	Convergence time	threshold
← ← ← ← ↓ ← ↓ → → ↑ ← ↓ ↓ ↓ ↓ ↑ ← ← ↓ ↓ ← ↓ ← ↑ ↑ ↓ ↓ ← → ↓ ↑ ↑ ← → ↓ ← ← ← ↓ → ↓ ↓ ↓ → ↓ → ↑ ↑ → ↑ ← ← ↓ ← → ↓ ↓ ↓ → ↓ ↓ → ↑ → ↑ ← ↓ ↓ ↓ ↓ → ↓ ← ↓ ↓ ↓ ↓ ↑ → ↑ ← ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ← ← ← ↑ ↑ → → → → → → ↓ ← ↓ ↓ ↑ ← → → ← ↑ ↑ ↑ ↑ ↑ → ↓ ↓ → ↓ ↑ ← ↓ ↑ ↑ → → → → → → ↓ ↓ ↓ ↓ ↑ → ↑ ← ↑ → ↑ → → → → ↓ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ → ↓ ↓ → → → → ↓ ↓ ↓ ↓ ↓ ← → → → → → → → → → → ↓ ↓ ↓ ← ← → → → → → → → → → → ↓ ↓ ← ← ← ↓ → → → ↓ → ↓ → → → ↓ ← ← ← → → → → → → → → → → → *	۳.۱۰۷۵۲	1
← ← ← ← ↓ ← ↓ → → ↑ ← ↓ ↓ ↓ ↓ ↑ ← ← ↓ ↓ ← ↓ ← ↑ ↑ ↓ ↓ ← → ↓ ↑ ↑ ← → ↓ ← ← ← ↓ → ↓ ↓ ↓ → ↓ → ↑ ↑ → ↑ ← ← ↓ ← → ↓ ↓ ↓ → ↓ ↑ → ↑ → ↑ ↓ ↓ ↓ ↓ ↓ → ↓ ← ↓ ↓ ↓ ↓ ↑ → → ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ← ← ← ↑ → → → → → → → ↓ ← ↓ ↓ ↑ ← → → → ↑ ↑ ↑ ↑ ↑ → ↓ ↓ → ↓ ↑ ← ↓ ↑ ↑ → → → → → → ↓ ↓ ↓ ↓ ↑ → ↑ ← ↑ → ↑ → → → → ↓ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ → ↓ ↓ → → → → ↓ ↓ ↓ ↓ ↓ ← → → → → → → → → → → ↓ ↓ ↓ ← ← → → → → → → → → → → ↓ ↓ ← ← ← ↓ → → → ↓ → ↓ → → → ↓ ← ← ← → → → → → → → → → → → *	۵.۲۶۱۴۴	0.1

	۸.۴۱۷۶۳	0.01
	۱۲.۶۵۲۶۱	0.001

--- | ← | ← | ← | ← | ↓ | ← | ↓ | → | → | ↑ | ← | ↓ | ↓ | ↓ | ↓ | --- | ↑ | ← | ← | ↓ | ↓ | ← | ↓ | ← | ↑ | ↑ | ↓ | ↓ | ← | → | ↓ | --- | ↑ | ↑ | ← | → | ↓ | ← | ← | ← | ↓ | → | ↓ | ↓ | ↓ | → | ↓ | --- | → | ↑ | ↑ | → | ↑ | ← | ← | ↓ | ← | → | ↓ | ↓ | ↓ | → | ↓ | --- | ↑ | → | ↑ | → | ↑ | ↓ | ↓ | ↓ | ↓ | ↓ | → | ↓ | ← | ↓ | ↓ | --- | ↓ | ↓ | ↑ | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | --- | ← | ← | ← | ↑ | → | → | → | → | → | → | → | ↓ | ← | ↓ | ↓ | --- | ↑ | ← | → | → | → | ↑ | ↑ | ↑ | ↑ | ↑ | → | ↓ | ↓ | → | ↓ | --- | ↑ | ← | ↓ | ↑ | ↑ | → | → | → | → | → | → | ↓ | ↓ | ↓ | ↓ | --- | ↑ | → | ↑ | ← | ↑ | → | ↑ | → | → | → | → | ↓ | ↓ | ↓ | ↓ | --- | ↓ | ↑ | ↑ | ↑ | ↑ | → | ↓ | ↓ | → | → | → | → | ↓ | ↓ | ↓ | --- | ↓ | ↓ | ↑ | → | → | → | → | → | → | → | → | → | → | → | ↓ | --- | ↓ | ↓ | ← | ← | → | → | → | → | → | → | → | → | → | → | ↓ | --- | ↓ | ← | ← | ← | ↓ | → | → | → | ↓ | → | ↓ | → | → | → | ↓ | --- | ← | ← | ← | → | → | → | → | → | → | → | → | → | → | → | * | ---	۱۳.۲۲۵۴۵	0.0001
--- | ← | ← | ← | ← | ↓ | ← | ↓ | → | → | ↑ | ← | ↓ | ↓ | ↓ | ↓ | --- | ↑ | ← | ← | ↓ | ↓ | ← | ↓ | ← | ↑ | ↑ | ↓ | ↓ | ← | → | ↓ | --- | ↑ | ↑ | ← | → | ↓ | ← | ← | ← | ↓ | → | ↓ | ↓ | ↓ | → | ↓ | --- | → | ↑ | ↑ | → | ↑ | ← | ← | ↓ | ← | → | ↓ | ↓ | ↓ | → | ↓ | --- | ↑ | → | ↑ | → | ↑ | ↓ | ↓ | ↓ | ↓ | ↓ | → | ↓ | ← | ↓ | ↓ | --- | ↓ | ↓ | ↑ | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | --- | ← | ← | ← | ↑ | → | → | → | → | → | → | → | ↓ | ← | ↓ | ↓ | --- | ↑ | ← | → | → | → | ↑ | ↑ | ↑ | ↑ | ↑ | → | ↓ | ↓ | → | ↓ | --- | ↑ | ← | ↓ | ↑ | ↑ | → | → | → | → | → | → | ↓ | ↓ | ↓ | ↓ | --- | ↑ | → | ↑ | ← | ↑ | → | ↑ | → | → | → | → | ↓ | ↓ | ↓ | ↓ | --- | ↓ | ↑ | ↑ | ↑ | ↑ | → | ↓ | ↓ | → | → | → | → | ↓ | ↓ | ↓ | --- | ↓ | ↓ | ↑ | → | → | → | → | → | → | → | → | → | → | → | ↓ | --- | ↓ | ↓ | ← | ← | → | → | → | → | → | → | → | → | → | → | ↓ | --- | ↓ | ← | ← | ← | ↓ | → | → | → | ↓ | → | ↓ | → | → | → | ↓ | --- | ← | ← | ← | → | → | → | → | → | → | → | → | → | → | → | * | ---	۱۷.۷۸۸۴۸	0.00001

--- | ← | ← | ← | ← | ↓ | ← | ↓ | → | → | ↑ | ← | ↓ | ↓ | ↓ | ↓ | --- | ↑ | ← | ← | ↓ | ↓ | ← | ↓ | ← | ↑ | ↑ | ↓ | ↓ | ← | → | ↓ | --- | ↑ | ↑ | ← | → | ↓ | ← | ← | ← | ↓ | → | ↓ | ↓ | ↓ | → | ↓ | --- | → | ↑ | ↑ | → | ↑ | ← | ← | ↓ | ← | → | ↓ | ↓ | ↓ | → | ↓ | --- | ↑ | → | ↑ | → | ↑ | ↓ | ↓ | ↓ | ↓ | ↓ | → | ↓ | ← | ↓ | ↓ | --- | ↓ | ↓ | ↑ | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | --- | ← | ← | ← | ↑ | → | → | → | → | → | → | → | ↓ | ← | ↓ | ↓ | --- | ↑ | ← | → | → | → | ↑ | ↑ | ↑ | ↑ | ↑ | → | ↓ | ↓ | → | ↓ | --- | ↑ | ← | ↓ | ↑ | ↑ | → | → | → | → | → | → | ↓ | ↓ | ↓ | ↓ | --- | ↑ | → | ↑ | ← | ↑ | → | ↑ | → | → | → | → | ↓ | ↓ | ↓ | ↓ | --- | ↓ | ↑ | ↑ | ↑ | ↑ | → | ↓ | ↓ | → | → | → | → | ↓ | ↓ | ↓ | --- | ↓ | ↓ | ↑ | → | → | → | → | → | → | → | → | → | → | → | ↓ | --- | ↓ | ↓ | ← | ← | → | → | → | → | → | → | → | → | → | → | ↓ | --- | ↓ | ← | ← | ← | ↓ | → | → | → | ↓ | → | ↓ | → | → | → | ↓ | --- | ← | ← | ← | → | → | → | → | → | → | → | → | → | → | * | ---	۲۳.۳۱۵۸۴	0.000001

با هر یک از مقادیر ۱، ۰.۱، ۰.۰۱، ۰.۰۰۱، ۰.۰۰۰۱، ۰.۰۰۰۰۱، ۰.۰۰۰۰۰۱ به عنوان ترشولد، الگوریتم value iteration به یک سیاست همگرا شد. هر چه مقدار ترشولد کوچکتر شد، الگوریتم در مدت زمان بیشتری سیاست همگرا شده را پیدا کرد. البته هیچ یک از این سیاست های همگرا شده، سیاست بهینه ای نبود که عامل را به خانه هدف برساند.

مقدار ترشولد باید نه خیلی بزرگ باشد که الگوریتم به سرعت همگرا شود و سیاست های ابتدایی را به عنوان سیاست همگرا شده و بهینه خروجی بدهد، نه باید خیلی کوچک باشد تا الگوریتم در مدت زمان خیلی زیادی همگرا بشود.

ب)

کد های مربوط به پیاده سازی این قسمت در فایل ***Codes.ipynb/html*** قسمت – **Problem 3 Part 3 - B** قرار دارد.

برای اجرا کد های این بخش لازم است تا تمام کد های ماقبل بخش اول و کد های این بخش را اجرا کنید.

زیر بخش ۱

آیا با سیاست بهینه بدست آمده در قسمت الف، عامل به خانه ی هدف می رسد؟

خیر. سیاست بهینه بدست آمده نمی تواند عامل را به خانه هدف برساند. علت این امر زیاد شدن تعداد استیت های مسئله نسبت به بخش قبل است. در نتیجه با مقادیر قبلی تعریف شده برای پاداش ها و مجازات ها نمی توان عامل را به خانه هدف رساند. افزایش تعداد استیت های مسئله باعث می شود تا در هنگام محاسبه ارزش یک استیت، با حلقه های طولانی تری برای تعیین ارزش استیت فعلی با توجه به ارزش استیت های بعدی مواجه شویم. در نتیجه اگر مقدار پاداش خانه هدف مقدار کمی نسبت به سایر پاداش ها و مجازات های تعریف شده در مسئله داشته باشد، ارزش خانه هدف در طی محاسبه ارزش یک استیت به دلیل آنکه فاصله و تعداد استیت های زیادی وجود دارد که از استیت فعلی به استیت نهایی برسیم، در یک مقدار discount factor به توان عدد زیادی ضرب می شود و ارزش خانه هدف کاهش می یابد. در نتیجه در یک استیت، حرکت در جهتی که ما را به خانه هدف نزدیکتر می کند، تبدیل به یک اکشن که باید دارای بیشترین ارزش در بین اکشن های مجاز یک استیت باشد، نمی شود.

زیر بخش ۲

مقدار پاداش خانه هدف را به نحوی تغییر دهید که مسیری که از خانه ی اول شروع می شود، با اتخاذ سیاست بهینه به خانه هدف برسد. مشاهدات خود در این قسمت را تحلیل و توجیه کنید.

برای حل مشکل مطرح شده، ما مقدار پاداش رسیدن به خانه هدف را از مقدار ۱۰۰ به مقدار ۱۱۰۰ تغییر می دهیم. سیاست بهینه جدید به شکل زیر می باشد:

```
optimal policy :

-------------------------------------------------------------
| ↓ | ↓ | ← | ↓ | ↓ | ↓ | ↓ | → | → | → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | → | → | → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | → | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | → | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | → | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| ↓ | → | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | → | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| ↓ | → | → | → | → | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | → | → | → | → | → | → | → | → | → | → | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| ↓ | ↓ | → | → | → | ↓ | → | ↓ | ↓ | ↓ | → | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | → | ↓ | → | → | → | → | → | → | → | → | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | → | → | → | → | → | → | → | → | → | → | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| → | → | → | → | ↓ | → | ↓ | ↓ | → | ↓ | → | ↓ | ↓ | ↓ | ↓ |
-------------------------------------------------------------
| ↓ | → | → | → | → | → | → | → | → | → | → | → | → | ↓ | ↓ |
-------------------------------------------------------------
| ↓ | → | → | → | → | → | → | → | → | → | → | → | → | ↓ | ↓ |
-------------------------------------------------------------
| ↓ | → | → | → | → | → | → | → | → | → | → | → | → | → | ↓ |
-------------------------------------------------------------
| → | → | → | → | → | → | → | → | → | → | → | → | → | → | * |
-------------------------------------------------------------
```

Figure ۱۲ سیاست بهینه

مشاهده می شود که در این سیاست عامل می تواند با شروع از خانه اول و دنبال کردن این سیاست به خانه هدف برسد.

در حالتی که پاداش خانه هدف برابر ۱۰۰ بود، ارزش اکثر استیت ها و حتی خیلی از استیت هایی که نزدیک خانه هدف بودند، مقداری بسیار کم و حتی مقداری زیر صفر داشتند. به علت کمبود پاداش خانه هدف و همچنین زیاد بودن استیت های این بخش، سیاست همگرا شده نمی توانست عامل را به خانه هدف برساند زیرا مقدار ارزش استیت-اکشن ها به خوبی باعث نمی شد تا در اکثر استیت ها، اکشنی که حرکت در آن جهت ما را به خانه هدف نزدیکتر می کند به عنوان سیاست بهینه در یک استیت انتخاب شود. در واقع کمبود مقدار پاداش خانه هدف باعث می شد تا در اکثر استیت ها، اکشنی که با انجام آن در آن استیت به خانه هدف نزدیکتر می شدیم، نسبت به سایر اکشن های مجاز در آن استیت مقدار بالاتری نداشته باشد.

ما این مشکل را با افزایش مقدار پاداش خانه هدف به ۱۱۰۰ حل کردیم. این کار باعث شد در هر استیت، مقادیر ارزش استیت-اکشن ها طوری مقداری دهی شود که در یک استیت، اکشنی ما را به خانه هدف نزدیکتر می کند، نسبت به سایر اکشن های مجاز در آن استیت ارزش بالاتری داشته باشد. با مقایسه مقادیر ارزش استیت ها در این حالت نسبت به حالت قبل(حالتی که پاداش خانه هدف برابر ۱۰۰ بود)، مشاهده می کنیم که هر چه به خانه هدف نزدیکتر می شویم، مقدار ارزش استیت ها با نسبت و اختلاف مناسبی افزایش پیدا می کند طوری که اکشن بهینه در هر استیت، اکشنی است که ما را به خانه هدف نزدیکتر می کند و ارزش بالاتری نسبت به سایر اکشن های مجاز در آن استیت دارد.

منابع

1. https://arxiv.org/pdf/2101.06286.pdf
2. https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/mdps-exact-methods.pdf
3. https://towardsdatascience.com/policy-and-value-iteration-78501afb41d2#:~:text=In%20Policy%20Iteration%2C%20at%20each,be%20the%20estimated%20state%20value.
4. https://www.baeldung.com/cs/ml-value-iteration-vs-policy-iteration