



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی برق

پروژه کارشناسی
گرایش مخابرات

ماشین های خودران -
با استفاده از یادگیری تقویتی

نگارش
محمد رضیئی فیجانی

استادان راهنما
دکتر وحید پوراحمدی و دکتر حمیدرضا امین داور

شهریور ۱۳۹۸

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

صفحه فرم ارزیابی و تصویب پایان نامه - فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تأیید و تصویب پایان نامه موسوم به فرم کمیته دفاع - موجود در پرونده آموزشی - را قرار دهید.

نکات مهم:

- نگارش پایان نامه/رساله باید به **زبان فارسی** و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت **پشت و رو(دورو)** بلامانع است و انجام آن توصیه می شود.

به نام خدا

تاریخ: شهریور ۱۳۹۸

تعهدنامه اصالت اثر



اینجانب **محمد رضیئی فیجانی** متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

محمد رضیئی فیجانی

امضا

نویسنده پایان نامه، در صورت تمایل میتواند برای پاسخگویی پایان نامه خود را به شخص
یا اشخاص و یا ارگان خاصی تقدیم نماید.

سپاس‌گزاری

نویسنده پایان‌نامه می‌تواند مراتب امتنان خود را نسبت به استاد راهنما و استاد مشاور و یا دیگر افرادی که طی انجام پایان‌نامه به نحوی او را یاری و یا با او همکاری نموده‌اند ابراز دارد.

محمدرضیٰ فغانی
شهریور ۱۳۹۸

چکیده

در این قسمت چکیده پایان نامه نوشته می‌شود. چکیده باید جامع و بیان‌کننده خلاصه‌ای از اقدامات انجام‌شده باشد. در چکیده باید از ارجاع به مرجع و ذکر روابط ریاضی، بیان تاریخچه و تعریف مسئله خودداری شود.

واژه‌های کلیدی:

کلیدواژه اول، ...، کلیدواژه پنجم (نوشتن سه تا پنج واژه کلیدی ضروری است)

فهرست مطالب

عنوان

صفحه

۱	یادگیری تقویتی با استفاده از gym	۱
۱-۱	معرفی مفاهیم یادگیری تقویتی	۲
۲-۱	معرفی OpenAI gym	۲
۱-۲-۱	مقدمه	۲
۲-۲-۱	نصب	۲
۲	معرفی یادگیری تقویتی	۳
۱-۲	مقدمه	۴
۱-۱-۲	جایگاه یادگیری تقویتی در یادگیری ماشین	۴
۲-۱-۲	وجه تمایز یادگیری تقویتی از دیگر الگوهای یادگیری ماشین	۵
۳-۱-۲	عامل و محیط	۶
۴-۱-۲	حالت	۷
۵-۱-۲	مشاهده پذیری ^۱	۸
۳	پیشنیازهای نصب و معرفی قسمت های مختلف	۱۰
۱-۳	نرم افزارهای کلی	۱۱
۲-۳	پیشنیازهای پایتون	۱۲
۳-۳	معرفی دقیق تر اجزای کلی	۱۳
۱-۳-۳	معرفی نرم افزار پری اسکن ^۲	۱۳
۲-۳-۳	فرمت های فایل های خروجی	۱۴
۳-۳-۳	نصب موتور متلب ^۳	۱۵
۴-۳	معرفی دقیق تر پیشنیازهای پایتون	۱۶
۱-۴-۳	بسته های کمکی	۱۶
۲-۴-۳	بسته gym	۱۶

^۱Observability

^۲PreScan

^۳Matlab Engine

۴	توضیح مختصری بر الگوریتم	۱۹
۴-۱	معرفی محیط شبیه سازی	۲۰
۴-۲	معرفی رابط برنامه نویسی برنامه و الگوریتم	۲۲
۴-۳	تعریف کردن پارامتر های یادگیری تقویتی	۲۶
۴-۳-۱	معرفی برخی توابع رابط برنامه نویسی برنامه	۲۷
۴-۳-۲	بررسی تابع <code>_next_observation</code> :	۳۲
۵	فنی	۳۹
۶	شبیه سازی و نتایج	۴۱
۶-۱	راه اندازی	۴۲
	منابع و مراجع	۴۴
	نمایه	۴۵
	فهرست اختصارات	۴۶
	واژه نامه انگلیسی به فارسی	۴۷
	واژه نامه فارسی به انگلیسی	۴۹

فهرست اشکال

شکل	صفحه
۱-۲	۴
۲-۲	۶
۳-۲	۷
۱-۳	۱۲
۲-۳	۱۳
۳-۳	۱۳
۴-۳	۱۴
۱-۴	۲۰
۲-۴	۲۱
۳-۴	۲۱
۴-۴	۲۹
۵-۴	۳۰
۶-۴	۳۲
۷-۴	۳۳
۸-۴	۳۴
۹-۴	۳۶
۱۰-۴	۳۸

⁴Observation⁵Reward

فهرست جداول

صفحه

جدول

۱-۳	توضیحات فرمت فایل خروجی	۱۵
۲-۳		۱۷
۱-۴	بررسی پارامترهای موجود در <code>env_dict</code>	۲۴
۲-۴	راهنمای توابع اصلی رابط برنامه‌نویسی برنامه	۲۶
۳-۴	راهنمای توابع کمکی رابط برنامه‌نویسی برنامه	۲۸
۴-۴	تعریف فضای مشاهده ^۶ و فضای حرکت ^۷ در پروژه	۲۸

^۶Observation Space

^۷Action Space

فهرست نمادها

نماد	مفهوم
\mathbb{R}^n	فضای اقلیدسی با بعد n
\mathbb{S}^n	کره n یکه بعدی
M^m	خمینه m -بعدی M
$\mathfrak{X}(M)$	جبر میدان‌های برداری هموار روی M
$\mathfrak{X}^1(M)$	مجموعه میدان‌های برداری هموار یکه روی (M, g)
$\Omega^p(M)$	مجموعه p -فرمی‌های روی خمینه M
\mathcal{Q}	اپراتور ریچی
\mathcal{R}	تانسور انحنای ریمان
ric	تانسور ریچی
L	مشتق لی
Φ	۲-فرم اساسی خمینه تماسی
∇	التصاق لوی-چویتای
Δ	لاپلاسین ناهموار
∇^*	عملگر خودالحاق صوری القا شده از التصاق لوی-چویتای
g_s	متر ساساکی
∇	التصاق لوی-چویتای وابسته به متر ساساکی
Δ	عملگر لاپلاس-بلترامی روی p -فرم‌ها

فصل اول

یادگیری تقویتی با استفاده از gym

۱-۱ معرفی مفاهیم یادگیری تقویتی

۲-۱ معرفی OpenAI gym

۱-۲-۱ مقدمه

پروژه gym از قوی ترین پروژه های Open AI^۱ می باشد.

۲-۲-۱ نصب

^۱<https://github.com/openai>

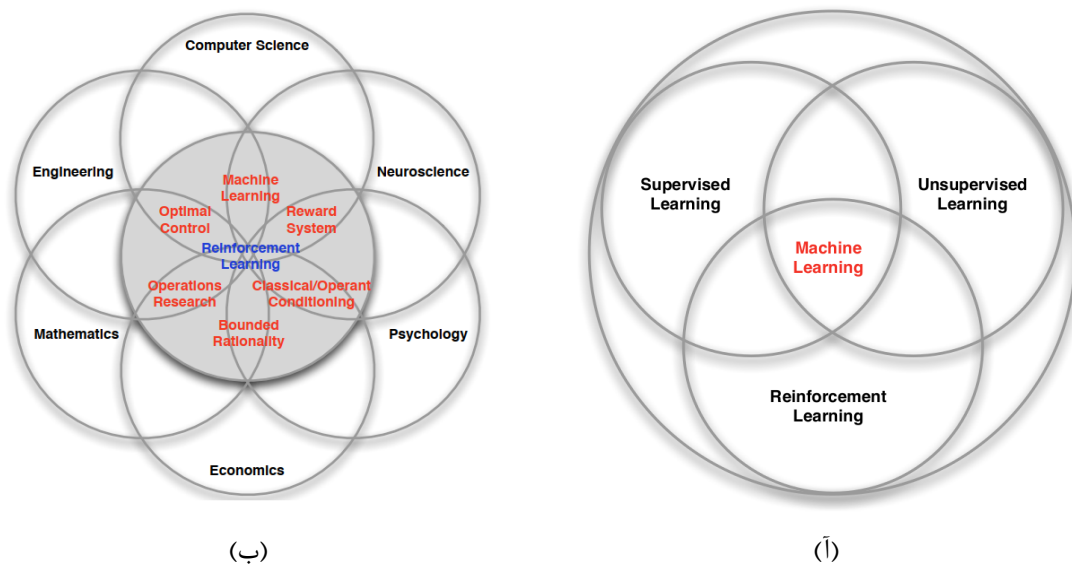
فصل دوم

معرفی یادگیری تقویتی

۱-۲ مقدمه

۱-۱-۲ جایگاه یادگیری تقویتی در یادگیری ماشین

بسیاری از صاحب نظران یادگیری ماشین^۱ را به سه دسته تقسیم می‌کنند: (۱) یادگیری با ناظر^۲ (۲) یادگیری بدون ناظر^۳ یا خوشه بندی^۴ (۳) یادگیری شبه ناظر^۵ در این میان، یادگیری تقویتی^۶ را بعضی ها دسته چهارم می‌دانند و بعضی دیگر آن را در دسته سوم قرار می‌دهند. بر اساس دسته‌بندی گروه دوم شکل ۱-۲(آ) رسم شده است. همچنین شکل ۱-۲(ب) کاربرد یادگیری تقویتی را در علوم مختلف نشان می‌دهد.



شکل ۱-۲:

¹Machine Learning²Supervised Learning³Unsupervised Learning⁴Clustering⁵Semi-Supervised Learning⁶Reinforcement Learning

۲-۱-۲ چه چیزی یادگیری تقویتی را با دیگر الگوهای یادگیری ماشین متمایز می‌کند؟

این سوال از آن جهت حایز اهمیت است که بیان می‌کند چرا ما به سراغ الگوی یادگیری تقویتی رفته‌ایم. پاسخ ملاحظات زیر است.

آ) هیچ ناظر^۷ وجود ندارد و صرفاً امتیازها وجود دارند.

ب) فیدبک^۸ همراه با تاخیر است و به صورت همزمان رخ نمی‌دهد.^۹

ج) مفهوم زمان واقعا مطرح است و یک ترتیب خاص از داده‌ها داریم. شکل ۲-۳ این توالی زمانی را نشان می‌دهد.

یادگیری تقویتی (RL^{۱۰}) بر اساس فرضیه امتیازها^{۱۱} پایه‌گذاری می‌شود.

تعریف ۲-۱-۱ (فرضیه امتیازها). همه اهداف می‌توانند براساس بیشینه کردن مقدار میانگین تجمعی امتیازها توصیف کرد.

ممکن است این عبارت کمی عجیب بنظر برسد اما در بسیاری از مسایل که به صورت برد و باخت و به نوعی دو حالت مطلوب و نامطلوب دارند، می‌توان در ساده‌ترین حالت مقدار $+1$ را برای برد و -1 را برای باخت در نظر گرفت.

نکته ۲-۱-۲. در برخی منابع بجای امتیاز از مفهوم هزینه^{۱۲} استفاده می‌کنند و هدف الگوریتم آن می‌شود که به سمتی حرکت کند که کمترین هزینه را داشته باشد. برای یک پارچه‌سازی این مفاهیم معمولاً یک علامت منفی برای این دو در نظر می‌گیرند یعنی :

$$r = -c \quad : \quad \text{هزینه} = -\text{امتیاز}$$

⁷Supervisor

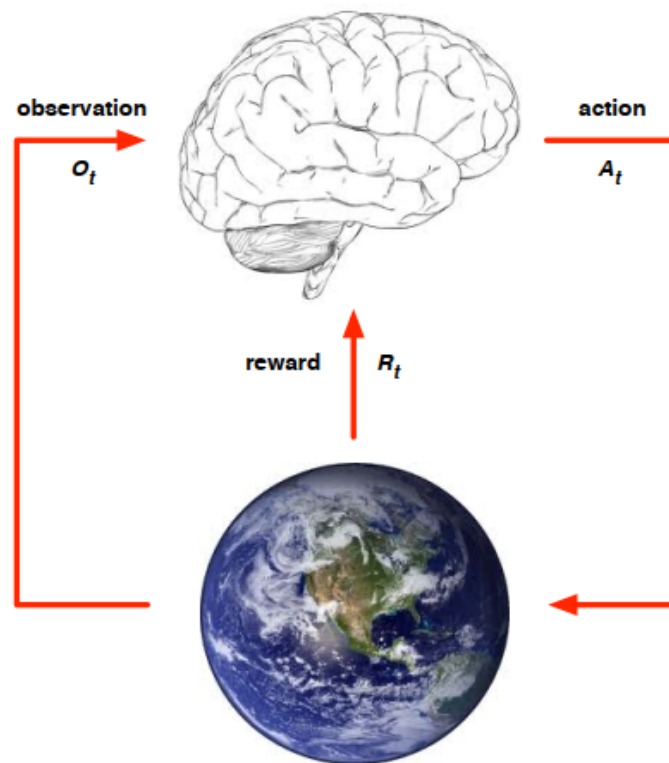
⁸Feedback

^۹در مورد علت تاخیر در ادامه توضیح داده خواهد شد.

¹⁰Reinforcement Learning

¹¹Reward Hypothesis

¹²Cost



شکل ۲-۲:

۳-۱-۲ عامل و محیط

این مفهوم بسیار مفهوم مهمی می‌باشد و بارها از آن در این پروژه یاد شده است. در مسایل یادگیری تقویتی یک **عامل**^{۱۳} وجود دارد که در یک **محیط**^{۱۴} در حال تعامل است. محیط می‌تواند محیط اطراف عامل باشد و یا هر چیزی که عامل با آن در تعامل است. [۱] این تعامل به این صورت است که عامل که در ابتدا یک حالت^{۱۵} اولیه دارد، یک حرکت^{۱۶} بر روی محیط در زمان t انجام می‌دهد. محیط مقدار حرکت در زمان t را دریافت می‌کند و سپس محیط در زمان $t + 1$ دو اطلاعات مهم را بر می‌گرداند. (آ) مشاهده (ب) امتیاز

شکل ۲-۲ و ۳-۲ این تعامل را نشان می‌دهد. لازم به ذکر است که در پایان هر مرحله^{۱۷} مقدار t یک واحد افزایش می‌یابد.

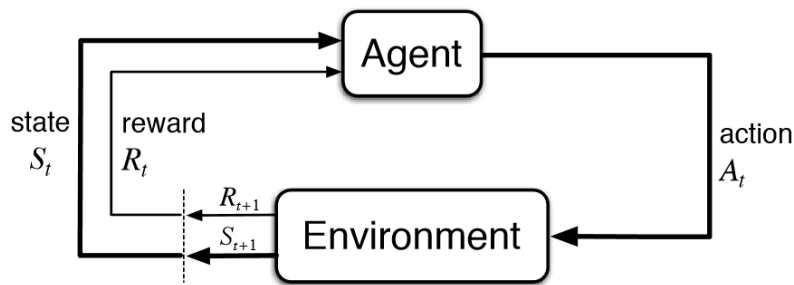
¹³ Agent

¹⁴ Environment

¹⁵ State

¹⁶ Action

¹⁷ Step



شکل ۲-۳:

۴-۱-۲ حالت

در بخش قبل تعریف مناسبی از حالت ارائه نشد. برای این تعریف ابتدا مفهوم تاریخچه^{۱۸} ارائه می‌شود و از روی آن حالت تعریف خواهد شد.

تعریف ۲-۱-۳ (تاریخچه). به سری شامل مشاهده، حرکت و امتیاز می‌باشد:

$$H_t = O_1, R_1, A_1, \dots, A_{t-1}, O_{t-1}, R_t$$

با این تعریف حالت را می‌توان به شکل زیر تعریف کرد.

تعریف ۲-۱-۴. حالت اطلاعاتی است که در محاسبات برای آن که در بعد چه اتفاقی بیافتد، استفاده می‌شود. به عبارت دیگر حالت تابعی از تاریخچه می‌باشد.

$$S_t = f(H_t)$$

دو نوع حالت وجود دارد.

آ) **حالت محیط**^{۱۹} که با علامت S_t^e نشان داده می‌شود. اطلاعات نهان محیط را نشان می‌دهد و معمولاً برای عامل به‌طور کامل دیده نمی‌شود. حتی اگر برای عامل مشاهده‌پذیر نیز باشد، ممکن است اطلاعات کاملاً بی‌ربطی را همراه داشته باشد.

ب) **حالت عامل**^{۲۰} که با علامت S_t^a نشان داده می‌شود. که برابر است با هر اطلاعاتی که عامل برای

¹⁸History

¹⁹Environment State

²⁰Agent State

رسیدن به حرکت بعدی با استفاده از الگوریتم های RL استفاده می کند.

بنابراین در تعریف ۲-۱-۴ مناسبتر است بجای واژه حالت از حالت عامل استفاده شود. بنابراین:

$$S_t^a = f(H_t)$$

یادداشت ۲-۱-۵. از این پس در سراسر این پایان نامه هر جا صحبت از حالت شد منظور همان حالت عامل است.

تعریف ۲-۱-۶. یک حالت S_t مارکوف^{۲۱} است اگر و تنها اگر:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

در یک حالت مارکوف^{۲۲}، آینده از گذشته مستقل است و فقط به زمان حال وابسته است. و این به این معناست که حالت از لحاظ آماری برای توصیف آینده کافی است.

نکته ۲-۱-۷. حالت محیط S_t^e مارکوف است. همچنین تاریخچه نیز مارکوف است.

۲-۱-۵ مشاهده پذیری

مشاهده پذیری کامل

عامل به طور مستقیم حالت محیط را مشاهده می کند. بنابراین در این حالت داریم:

$$O_t = S_t^a = S_t^e$$

بنابراین در این حالت عبارت های زیر با یکدیگر برابر هستند.

$$\text{حالت اطلاعاتی} = \text{حالت محیط} = \text{حالت عامل}$$

²¹Markov

²²Markov State

^{۲۳} به صورت رسمی، این فرایند یک روند تصمیم‌گیری مارکوف^{۲۵} (MDP) می‌باشد. [۲]

مشاهده پذیری جزئی

عامل به‌طور غیر مستقیم محیط را مشاهده می‌کند.

نمونه ۸-۱-۲. یک ربات با دید دوربین نمی‌تواند موقعیت مطلق را اعلام کند.

نمونه ۹-۱-۲. یک اتومبیل با سنسور تشخیص فاصله نمی‌تواند اطلاعاتی مانند نوع ماشین و قیمت آن را تشخیص دهد.

^{۲۳} حالت اطلاعاتی^{۲۴} مفهومی مانند حالت مارکوف دارد.

^{۲۵} Markov Decision Process

فصل سوم

پیشنهاد های نصب و معرفی قسمت های مختلف

۳-۱ نرم افزارهای کلی

در این پروژه از جهت آنکه نسخه قبلی و پیشینی برای آن نبوده است، به ناچار می‌بایست که کد آن از صفر تا صد آن به صورت دستی نوشته شود. از این‌رو، پیچیدگی‌های بسیار فراوان را به طور خاص در پی داشت. ابزارهای زیادی نیز بنابه شرایط در آن استفاده شد که ارتباط بین آن ابزارها و اجزاء بر این پیچیدگی پیاده سازی طرح افزوده بود.

ابزارهای اصلی و کلی که در این پروژه استفاده شده بود، عبارتند از:

- نرم افزار پری اسکن ، نسخه 8.5.0

- نرم افزار متلب^۱ ، نسخه R2017b

- زبان برنامه نویسی پایتون ، نسخه 3.6.9

بنابراین برای راه اندازی مجدد کد این پروژه لازم است که موارد بالا روی کامپیوتر شخص به صورت کامل نصب باشد.

همچنین لازم به ذکر است که برخی ابزارات دیگر نیز در این پروژه استفاده شده است که احتمالاً با نصب موارد بالا دیگر نیازی به نصب آن‌ها به صورت جداگانه نیست. هدف این ابزارها ایجاد اتصال بین اجزای اصلی گفته شده است. این گروه شامل موارد زیر هستند:

- سیمولینک^۲ ، جهت اتصال بین متلب و پری اسکن

- شبکه UDP^۳ ، جهت اتصال داده های پویا^۴ بین پایتون و سیمولینک

- موتور متلب ، جهت اتصال داده های ساکن^۵ بین پایتون و سیمولینک

در این فصل جزئیات بیشتری در مورد لزوم و دلیل استفاده از این ابزارها بررسی می‌شود.

¹ Matlab

² Simulink

³ برای این منظور از ماژول socket در پایتون استفاده شده است.

⁴ Dynamic Data

⁵ Static Data



شکل ۳-۱: تقسیم‌بندی وظایف اصلی کد پایتون

۳-۲ پیشنیازهای پایتون

یادداشت ۳-۲-۱. کد پایتون در این پروژه شامل دو قسمت کلی زیر می‌شود. این دو دسته در شکل ۳-۱ مشخص هستند.

۱. دسته اول مربوط به آن بخش از پروژه است که وظیفه اصلی آن ارتباط پیدا کردن با محیط متلب و پری اسکن و ایجاد یک نوع واسط کاربری است. گرفتن و فرستادن اطلاعات مخصوص این قسمت است.

۲. دسته دوم با محیط و نحوه ارتباط آن کاری ندارد و تمرکز خود را بر روی الگوریتم خود که در این جا از الگوریتم‌های یادگیری تقویتی استفاده شده است، قرار داده است.

دسته اول (سمت چپ تصویر ۳-۱) به پکیج‌های زیر احتیاج دارد:

- numpy
- os و time
- matlab.engine
- socket
- pandas
- gym

اگر از آناکوندا^۶ برای پایتون استفاده می‌کنید غیر از دو بسته gym و matlab.engine به صورت پیش‌فرض نصب شده اند در صورت عدم نصب آن‌ها را با استفاده از pip^۷ می‌توان نصب کرد. بسته gym که در این فصل به تفصیل در مورد آن بحث شده است، به راحتی با همان دستور pip نصب می‌شود. اما نصب matlab.engine یا همان موتور متلب متفاوت است و نمی‌توان آن را نیز به همان روش نصب کرد.

دسته دوم شامل بسته‌های زیر است:

^۶Anaconda

^۷مثلاً بسته numpy را با استفاده از دستور pip install numpy نصب می‌توان کرد.

• gym[atari] یا gym[all]

• tensorflow

• stable-baseline

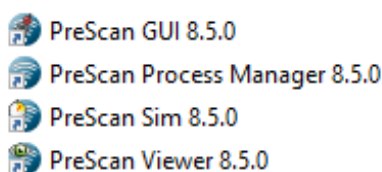
این بسته ها در لایه الگوریتم استفاده شده است. (در مورد این لایه در فصل ۴ بیشتر صحبت خواهد شد). هر سه تای این بسته ها با همان دستور pip به راحتی نصب می شوند.

۳-۳ معرفی دقیق تر اجزای کلی

در این قسمت میخواهیم سه نرم افزار کلی این پروژه را از نگاهی نزدیک تر بشناسیم که عبارتند از :
(۱) نرم افزار پری اسکن (۲) متلب (۳) پایتون

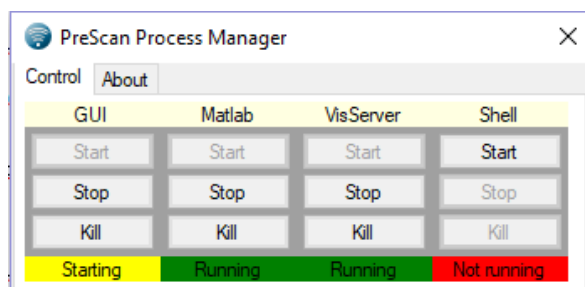
۱-۳-۳ معرفی نرم افزار پری اسکن

پس از دانلود و نصب نسخه 8.5.0 این نرم افزار چهار آیکون مانند شکل ۲-۳ به محیط دسکتاپ اضافه می کند. اصلی ترین آن ها PreScan Proccess Manager 8.5.0 نام دارد.



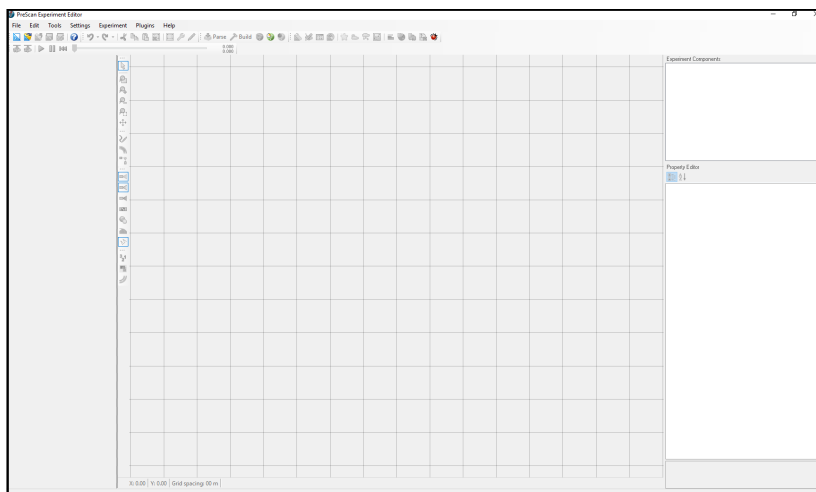
شکل ۲-۳: آیکون های اضافه شده بر روی محیط دسکتاپ پس از نصب پری اسکن

با انتخاب آن صفحه ای مانند زیر باز می شود.



شکل ۳-۳: پنل مدیریت نرم افزار پری اسکن

این پنجره شامل گزینه های زیر است:



شکل ۳-۴: صفحه گرافیکی محیط پری اسکن

• Matlab

• GUI

• Shell

• VisServer

برای ایجاد یک محیط جدید باید GUI را استارت کرد. پس از مدتی صفحه ای مانند شکل ۳-۴ باز می شود.

پس از ایجاد مدل ها و ذخیره آن، فایل های **.pex و **.pb و *_cs.slx ساخته می شود.^۸ جهت استفاده از فایل سیمولینک باید در شکل ۳-۳ متلب را استارت کنید.

نکته ۳-۳-۱. برای اجرای فایل های سیمولینک خروجی، لازم است که متلب را فقط و فقط با استفاده از نرم افزار پری اسکن و با استفاده از پنل مدیریت نرم افزار معرفی شده در شکل ۳-۳ باز شود. در صورتی که به صورت مستقیم این کار انجام شود، به مشکل منتهی می شود.

دو قسمت دیگر نیز در شکل ۳-۳ وجود دارد که نیازی به استارت کردن آن ها نیست و خودشان در صورت لزوم به صورت خودکار فراخوانی می شوند.

۳-۳-۲ فرمت های فایل های خروجی

نرم افزار پری اسکن پس از ایجاد یک محیط جدید، فایل ها و پوشه های بسیار زیادی را ایجاد می کند. اما در خارج آن پوشه ها ۳ فایل وجود دارد که پسوند آن ها **.pex و **.pb و *_cs.slx می باشد.

^۸علامت * به معنای یک اسم مشترک در این سه فایل استفاده شده است.

علامت ** همان اسم پروژه ای است که ایجاد کرده ایم. هر یک از این فایل ها به یک بلوک از شکل ۳-۳ مربوط می شود.

فرمت فایل	توضیحات
** .pex	این فایل مربوط به اولین بلوک شکل ۳-۳ است و ارتباط مستقیم با GUI دارد. برای تغییر محیط گرافیکی باید این فایل را باز کرد.
** .pb	این فایل برخی از اطلاعات فایل ** .pex را در اختیار دارد و با تغییر آن فایل این فایل نیز عوض می شود. این فایل حاوی اطلاعات استاتیک محیط ایجاد شده است و مهم ترین کاربرد آن در بلوک موتور متلب که در شکل ۳-۳ نشان داده شده است می باشد. پایتون از طریق این فایل این اطلاعات را دریافت می کند.
**_cs.slx	این فایل سیمولینک است که برای کار کردن با آن باید از پنل مدیریت شکل ۳-۳ استفاده کرد. این فایل پس از ایجاد از فایل ** .pex مستقل می شود. این فایل خود قابلیت تغییر دارد و می توان بلوک های آن را در محیط سیمولینک تغییر داد و بلوک های دیگری به آن افزود. در صورتی که فایل ** .pex تغییر کند، این امکان را نیز دارد که از داخل خود سیمولینک با فشردن دکمه ای این تغییرات جدید اعمال شود بدون آن که به تغییرات خود کاربر لطمه ای وارد شود. در این پروژه این فایل، تغییرات بسیاری را تجربه کرد.

جدول ۳-۱: توضیحات فرمت فایل خروجی

جدول ۳-۱ توضیحات لازم را جهت آشنایی با این خروجی ها آورده است. همچنین در بخش ۳-۳ در مورد فایل *_cs.slx توضیحات دقیق تری در مورد جزئیات آن گفته خواهد شد.

۳-۳-۳ نصب موتور متلب

برای نصب موتور متلب ابتدا نیاز است که به متلب به طور کامل در سیستم نصب باشد. پس از نصب متلب، محیط Command prompt (admin) را باز کنید و با توجه به نسخه و محل نصب متلب خود به آدرس زیر بروید.

```
<matlabroot>\extern\engines\python
```

مثلا برای Matlab R2017b که در محل پیش فرض خود نصب شده باشد این کار با استفاده از دستور زیر انجام می شود.

```
cd C:\Program Files\MATLAB\R2017a\extern\engines\python
```

در این پوشه یک فایل به نام setup.py موجود می باشد. این فایل را با استفاده از دستور

```
python setup.py install
```

در همان محیط cmd اجرا کنید.

یادداشت ۳-۲-۲. توجه داشته باشید که باید نسخه متلب و پایتون شما باید با یکدیگر سازگار باشند. برای بررسی این موضوع اگر فایل setup.py را با اسنفاده از یک ادیتور باز کنید، یک آرایه به نام supported_versions در آن خواهید دید. مقادیر این آرایه، نسخه هایی از پایتون را نشان می دهد که توسط نسخه متل شما پشتیبانی میشود مثلاً در این مورد، با توجه به خط زیر نسخه های ۲/۷، ۳/۴، ۳/۵ و ۳/۶ پایتون پشتیبانی می شود. در غیر این صورت باید نسخه سازگار متلب و یا پایتون را نصب کنید.

```
_supported_versions = ['2.7', '3.4', '3.5', '3.6']
```

۴-۳ معرفی دقیق تر پیشنهاد های پایتون

۱-۴-۳ بسته های کمکی

این بسته ها نقش حیاتی ندارند و برای برخی از موارد استفاده شده اند. این موارد در جدول ۲-۳ آمده است.

۲-۴-۳ بسته gym

معرفی

بسته gym که توسط OpenAI توسعه یافته است. این ابزار فوق العاده این امکان را برای محققین علوم کامپیوتر حرفه ای و یا آماتور فراهم می کند که انواع الگوریتم های یادگیری تقویتی (RL) را بر روی کار خود تست کنند. همچنین پتانسیل این را دارد که محققین محیط خود را بر روی این بسته توسعه دهند. هدف از ایجاد این بسته، استاندارد سازی محیط و نوعی نقطه تراز^{۱۲} برای پژوهش های RL محسوب می شود.^[۳]

در حقیقت می توان این بسته را در وسط شکل ۱-۳ جای داد. جایی که لایه محیط و لایه الگوریتم^{۱۳}

¹²Bechmark

¹³Algorithm

نام بسته	دلیل استفاده	روش نصب
numpy	ایجاد ماتریس برای فضای حرکت و فضای مشاهده	pip install numpy
time	جهت ایجاد تاخیر و سقف زمانی ^۹	pip install time
os	برای بستن پنجره های باز شده پس از اجرا	pip install os
pandas	برای چاپ اطلاعات آماری امتیاز های بدست آمده در پایان هر اپیزود ^{۱۰}	pip install pandas
socket	برقراری ارتباط با متلب و فرستان و دریافت کردن داده های پویا	pip install socket
tensorflow	برای لایه الگوریتم و استفاده از الگوریتم های یادگیری تقویتی عمیق ^{۱۱}	pip install tensorflow

جدول ۳-۲:

به یک دیگر می رسند.

این بسته محیط هایی از پیش ساخته شده دارد. نام این بسته ها در لیست زیر آمده است. ^{۱۴} اکثر این محیط ها نوعی بازی هستند که عامل سعی در یادگیری آن محیط ها دارد.

- Pong-v0
- MsPacman-v0
- SpaceInvaders-v0
- Seaquest-v0
- LunarLanderV2
- Reacher-v2
- FrozenLake-v0
- CartPole-v0
- Pendulum-v0
- MountainCar-v0
- MountainCarContinuous-v0
- BipedalWalker-v2
- Humanoid-V1
- Riverraid-v0
- Breakout-v0

نصب

برای نصب نسخه کمینه این نرم افزار با همان روش pip به راحتی می توان نرم افزار مورد نظر را نصب کرد. [۴] این نسخه کمینه برای لایه محیط کافی می باشد. اما اگر بخواهیم بخش الگوریتم را با استفاده از کتابخانه های دیگری مانند stable-baseline نوشت. نیازمند نسخه جامع تری از gym می باشد.

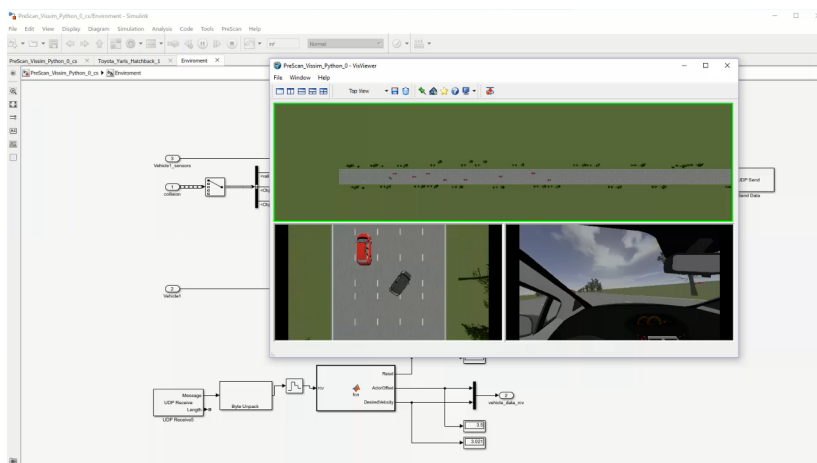
^{۱۴} جدول کامل در سایت <https://github.com/openai/gym/wiki/Table-of-environments> قرار دارد.

یادداشت ۳-۴-۱. پیشنهاد می شود برای نصب نسخه کامل gym و stable-baseline از لینوکس بجای ویندوز استفاده کنید. زیرا در نصب برخی بسته ها ممکن است با مشکل روبرو شوید.

برای نصب کامل این بسته از دستور `pip install gym[all]` استفاده کنید. ممکن است در نصب mujoco به مشکل برخوردید در این صورت دستور `pip install gym[atari]` استفاده کنید. اگر موفق به نصب این بسته نشدید می توانید مراجل نصب آن را با استفاده از [۳] مراجعه کنید.

فصل چهارم

توضیح مختصری بر الگوریتم



شکل ۴-۱: محیط شبیه سازی

در فصل ۲ در مورد مفاهیم یادگیری تقویتی بحث شد. مهمترین مفاهیم عبارتند از:

- | | | |
|---------|--------------|-----------------|
| ۱. محیط | ۳. حالت محیط | ۵. امتیاز |
| ۲. عامل | ۴. حالت عامل | ۶. مشاهده پذیری |

هدف در این پروژه این بود که یک ماشین خودران^۱ با استفاده از الگوریتم های یادگیری تقویتی ساخته شود. جزییات تئوری الگوریتم و جزییات فنی پروژه به ترتیب در بخش های ۲ و ۳ آورده شده‌اند. در این بخش به شبیه سازی و جزییات کار و تعریف پارامتر های این پروژه پرداخته می‌شود.

۴-۱ معرفی محیط شبیه سازی

در ابتدا محیط شبیه سازی را معرفی می‌کنیم. جزییات فنی این محیط در ۳ و ۴ و همچنین نحوه راه‌اندازی آن در بخش ۶-۱ به صورت کامل مورد بحث قرار گرفته است. اگر آن محیط را باز کنید محیط مانند شکل ۴-۱ باز خواهد شد. این محیط دو آبجکت مهم دارد؛ (آ) ماشین (اتومبیل) (ب) جاده (شکل ۴-۲) چیزی که اهمیت دارد اندازه ها و نحوه تعریف محدوده هاست. شکل ۴-۳ اندازه‌ها و محدوده ها را مشخص کرده است. شکل ۴-۳(ب) نشان می‌دهد که این محدوده ها کاملاً بر روی یک دیگر منطبق نیستند. دلیل اصلی این موضوع عدم اهمیت تطبیق دقیق این دو می‌باشد. در بخشی که پشت ماشین قرار دارد این محدوده از ۴- (کمی بیشتر از اندازه عرض لاین ها) شروع می‌شود. زیرا نیازی نیست بیشتر از این مقدار ماشین مورد بررسی به عقب برود تا متوجه شویم اشتباه در حال رفتن است. درحقیقت این

¹Autonomous Vehicle



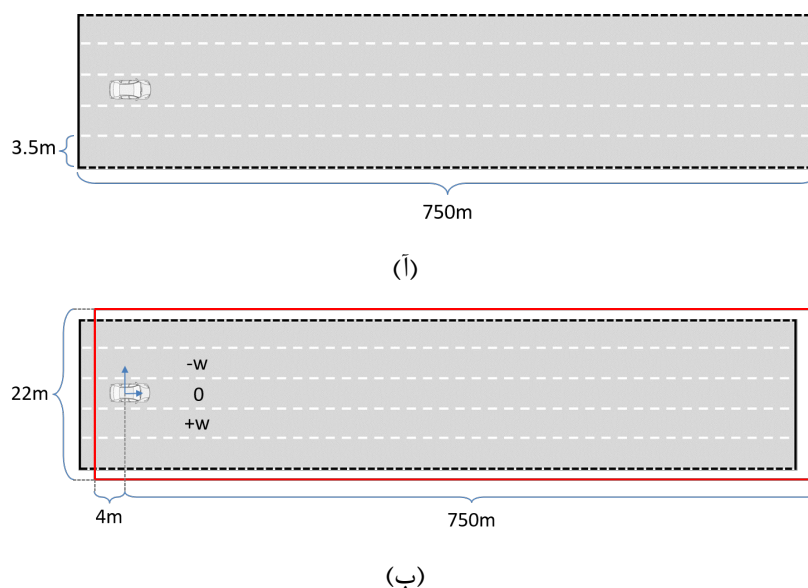
شکل ۲-۴:

مورد کمک می‌کند تا تعداد مرحله‌ها را در هر اپیزود اشتباه کاهش یابد. بخش‌های کناری نیز از ۱۱- تا ۱۱+ محدود شده‌اند (بیشتر از عرض خود جاده) تا اگر نوسانی یافت به ماشین این اجازه داده شود تا به مسیر اصلی برگردد.

یادداشت ۱-۱-۴. ماشین در مبدا صفحه قرار دارد. از این رو اعداد منفی نسبت به همین ماشین نیز سنجیده می‌شوند.

۳ مقدار $+w$ ، $-w$ و 0 که در شکل ۳-۴ (ب) بر روی جاده نوشته شده است در حقیقت مرتبط با بحث فنی ماجرا می‌باشد اما مفهوم آن این است که عامل مورد بررسی می‌تواند این سه لاین را به عنوان حرکت اختیار کند. در حقیقت می‌توان آن‌ها را به عنوان اسم برای هر لاین در نظر گرفت. در مورد حرکت بیشتر صحبت خواهد شد.

یادداشت ۲-۱-۴. راه‌اندازی این محیط کمی دردسر خواهد داشت از این‌رو نیاز است پیش از راه‌اندازی بخش ۱-۶ به‌طور دقیق مطالعه شود.



شکل ۳-۴:

۲-۴ معرفی رابط برنامه‌نویسی برنامه و الگوریتم

در این پروژه دو الگوریتم DQN^2 و $A2C^3$ بهتر از سایر الگوریتم‌ها عمل کردند اما در نهایت با توجه به آزمایش‌ها و ملاحظات که انجام شد، الگوریتم DQN از لحاظ سرعت همگرایی بهتر از الگوریتم $A2C$ پاسخ داد. بنابراین صرفاً بر روی این الگوریتم بحث خواهد شد.

```

1  import gym, gym_prescan
2
3  from stable_baselines.common.vec_env import DummyVecEnv
4  from stable_baselines.deepq.policies import MlpPolicy
5  from stable_baselines import DQN
6
7  save_load = "deepq_prescan"
8  env_dict = {
9      'id':      'prescan-without-matlabengine-v0',
10     'verbose': True,
11     'host':    '172.21.217.140',
12     # 'delay': 1,
13     'nget':    150
14 }
15
16 env = gym.make(**env_dict)
17 env = DummyVecEnv([lambda: env])
18 model = DQN(MlpPolicy, env, verbose=1, gamma=0.8,
19             prioritized_replay=True)
19 print('Model created!')
20 try:
21     model.learn(total_timesteps=50000)
22 except:
23     print('Error!')
24 model.save(save_load)

```

این کد بخش آموزش^۴ را نشان می‌دهد. بخش تست^۵ در تمامی الگوریتم‌ها مشابه یک دیگر است و از جایی که مدل تعریف می‌شود (در اینجا خط ۱۸) شروع خواهد شد.

²Deep Q-Learning Network

³Synchronous Actor Critics

⁴Train

⁵Test

بخش تست در تمامی الگوریتم ها کد زیر است.

```

1 model = DQN.load(save_load)
2 obs = env.reset()
3 while True:
4     print('-----TEST-----')
5     action, _states = model.predict(obs)
6     obs, rewards, dones, info = env.step(action)
7     env.render()

```

از روی چند خط آغازین کد DQN می‌توان دریافت که این کد با استفاده از gym [۴] و stable-baseline [۵] نوشته شده است.

بخش مهم بعدی متغیری از جنس دیکشنری به نام env_dict است. این متغیر برای ساختن متغیر env در دستور env = gym.make(**env_dict) به کار می‌رود. ^۶ توضیح این متغیر و اجزای آن در جدول ۱-۴ آمده است.

همان‌طور که در جدول ۱-۴ توضیح داده شده است؛ دو متغیر nget و delay هر دو از جنس تاخیر می‌باشند. محل تاخیر در تابع step می‌باشد. کد زیر محل تاخیر را نشان می‌دهد.

```

1 def step(self, action):
2     self.send(action)
3     # ----- BEGIN DELAY -----
4     if self.delay > 0 :
5         sleep(self.delay)
6     for _ in range(self.nget):
7         self.render_()
8         if self.done:
9             break
10    # ----- END DELAY -----
11    observation = self._next_observation()
12    reward = self.calc_reward()
13    done = self.done
14    info = {'Collision':self.collision
           , 'Position':self.agent['data']['Position']}

```

این کد که در حقیقت هسته اصلی ^۷ تابع step می‌باشد. در بین محدوده مشخص شده، تاخیر صورت

^۶ متغیر env در حقیقت نقش محیط را در الگوریتم دارد.

^۷ این کد از آن جهت که کاملاً با تابع اصلی برابر نمی‌باشد، واژه "هسته اصلی" برای آن در نظر گرفته شده است. تفاوت

متغیر	توضیحات
id	در این پروژه این متغیر دو حالت بیشتر ندارد که هر دو از جنس رشته هستند. اگر این کد با استفاده از موتور متلب استفاده شود، 'prescan-v0' خواهد بود و اگر از موتور متلب استفاده نشده باشد مقدار آن 'prescan-without-matlabengine-v0' خواهد بود. این متغیر مقدار پیش فرض ندارد.
verbose	این متغیر که از جنس بولین می باشد، در صورتی که یک باشد اطلاعات جامعی را در هر مرحله را چاپ می کند. علاوه بر آن اطلاعات آماری امتیازهای بدست آمده در پایان هر اپیزود را نیز چاپ می کند. به طور کلی اجازه گزارش دادن و ندادن اطلاعات درونی الگوریتم توسط این متغیر کنترل می شود.
host	این متغیر برای اتصال شبکه بین دو کامپیوتر به کار می رود و در حقیقت IP کامپیوتری است که ویندوز بر روی آن نصب است. مقدار پیش فرض این متغیر 'localhost' می باشد.
delay	همان طور که مشخص است این متغیر مقدار تاخیر را مشخص می کنم و مورد کاربرد آن لحظه ای است که action در تابع step فرستاده شده است و پس از گذشت مقداری تاخیر برحسب ثانیه سعی در دریافت اطلاعاتی مانند مشاهده بعدی و محاسبه امتیاز و ... داریم. مقدار پیش فرض این متغیر نیز صفر است.
nget	این متغیر نیز به نوعی متفاوت تاخیر را شکل می دهد. این متغیر از نوع عدد صحیح می باشد و هنگامی که مقدار آن ۱۵۰ است یعنی محل تاخیر، ۱۵۰ بار داده ها را دریافت می کند و مقدار آن ها را می خواند. در حالت عادی تا پایان ۱۵۰ امین دریافت هیچ کاری نمی کند مگر این که مقدار done برابر یک شود؛ در این صورت حلقه را متوقف کرده و باقی عملیات را انجام می دهد. مقدار پیش فرض این متغیر یک می باشد.
experimant_name	این متغیر مربوط به تنظیمات موتور متلب می باشد و به صورت عادی نیازی به تغییر مقدار پیش فرض آن نیست.
close_window	این پارامتر در صورتی قابل اجراست که کد پایتون و نرم افزار پری اسکن هر دو بر روی یک کامپیوتر باشند و وظیفه آن این است که محیط گرافیکی را پس از اجرا شدن کد می بندد و مقدار پیش فرض آن صفر می باشد.

جدول ۴-۱: بررسی پارامتر های موجود در env_dict

می‌گیرد. همان طور که مشخص است این تاخیر بین فرستادن حرکت و محاسبه پارامترهایی مانند امتیاز و مشاهده (حالت) می‌باشد.

یادداشت ۴-۲-۱. علت اصلی وجود تاخیر، مهلت دادن به عامل برای انجام حرکت است.

دو متغیر $nget$ و $delay$ هر دو وظیفه ایجاد تاخیر دارند که نحوه ایجاد این تاخیر با یک‌دیگر کاملاً متفاوت است. همچنین این امکان نیز وجود دارد به صورت ترکیبی نیز این تاخیر را ایجاد کرد. هر کدام از این روش‌ها مزایا و معایب خاص خود را دارند.

مزیت مهم استفاده از $nget$ این است که به صورت مداوم در حال دریافت اطلاعات از محیط شبیه‌سازی است. بنابراین در صورت رخ دادن اتفاق خاصی مانند تصادف کردن و یا از مسیر خارج شدن می‌تواند آن را به موقع تشخیص دهد و تصمیمات لازم را انجام دهد. در صورتی که در زمانی که تاخیر ناشی از $delay$ است، عملاً در آن مدت ارتباط با محیط شبیه سازی قطع شده است و ممکن است رخ دادن موارد گفته شده یا بسیار دیر متوجه شود و یا اصلاً متوجه نشود.

به طور مثال در صورتی که دو ماشین با یکدیگر تصادف داشته باشند؛ اگر این رخداد سریعاً تشخیص داده نشود، در این صورت احتمال دارد این دو ماشین از روی یک دیگر عبور کنند! و پس از عبور کردن این اطلاعات دریافت شود و برخوردی تشخیص داده نشود. از آن جا که برخورد بیشترین میزان تاثیر در امتیاز دارد؛ بنابراین، این اتفاق تاثیرات خیلی مخربی می‌تواند بر الگوریتم بگذارد.

عیب اصلی روش $nget$ نیز این است که یک مقدار مشخص ندارد و به پارامترهایی از جمله سرعت شبکه نیز وابسته است. بنابراین اگر از یک شبکه به شبکه دیگر منتقل شود می‌تواند مقدار کاملاً متفاوتی به خود گیرد که شاید مطلوب نباشد. اما به راحتی با عوض کردن مقدار این متغیر در لایه الگوریتم می‌توان این مشکل را حل کرد. بنابراین توصیه می‌شود از این متغیر استفاده شود.

به کد **DQN** برگردیم. خط ۱۸ این کد مدل را می‌سازد. چیزی که اهمیت دارد این است که پارامتر γ چه مقداری انتخاب شود. در نسخه نهایی این مقدار روی 0.8 تنظیم شده است. در مورد این متغیر در بخش ۲ و در مراجع [۲] و [۱] بحث شده است. در ابتدا این متغیر مقدار پیش فرض 0.99 را داشت. پس از بررسی‌های انجام شده این مقدار به 0.8 کاهش یافت.

آن با کد اصلی برخی عملیات است که مرتبط با چاپ شدن اطلاعات در حال اجرا می‌باشد که به مقدار $verbose$ مرتبط می‌شود.

نام تابع	توضیحات
<code>--init--</code>	این تابع علاوه بر تنظیم کردن برخی پارامترهای مرتبط به کلاس، اتصال کد پایتون به نرم افزار متلب را نیز برعهده دارد. همچنین تعیین فضای مشاهده و فضای حرکت نیز برعهده این بخش می باشد.
<code>step</code>	این تابع همان مرحله است که در بخش ۲ مطرح شد. محل اصلی اجرای این تابع در داخل یک حلقه متناهی می باشد. این تابع <code>action</code> را به صورت ورودی می گیرد و متغیرهایی مانند <code>observation</code> ، <code>reward</code> ، <code>done</code> و <code>info</code> را خروجی می دهد. در مورد نحوه محاسبه این خروجی ها صحبت خواهد شد.
<code>reset</code>	این تابع <code>env</code> ، را ریست می کند و به عنوان خروجی حالت اولیه را برمی گرداند. موارد استفاده از این تابع معمولاً در اول کد و در آخر هر اپیزود می باشد. آخر هر اپیزود هنگامی فرا می رسد که متغیر <code>done</code> که یکی از خروجی های تابع <code>step</code> است، یک شود.
<code>render</code>	این تابع به صورت معمول کارهای گرافیکی را برعهده دارد. اما از آنجایی که عمل در پس زمینه طرح وجود دارد، پس کار اصلی آن گرفتن داده ها و منظم کردن آن ها می باشد. برای این کار از یک تابع کمکی به نام <code>render_</code> استفاده می کند.

جدول ۴-۲: راهنمای توابع اصلی رابط برنامه نویسی برنامه

۴-۳ تعریف کردن پارامترهای یادگیری تقویتی

منظور از پارامترهای یادگیری تقویتی از متغیرهای حرکت و حالت و امتیاز و ... تا تعریف برخی توابع می باشد. ابتدا کلیات توابع را بررسی کنیم و سپس وارد جزئیات آن پارامترها می شویم.

توابع استفاده شده به دو دسته تقسیم می شوند. (آ) توابع اصلی (ب) توابع فرعی یا کمکی.

توابع اصلی آن دسته از توابعی هستند که مختص به کتابخانه `gym` هستند و قرار دادن آن ها به شکل صحیح آن، اجباری است. توابع کمکی آن دسته از توابعی هستند که در این توابع نقش های مشخصی را ایفا کردند ولی استفاده کردن از آن ها اجباری نداشته است.

یادداشت ۴-۳-۱. در صورت لزوم کاربر می تواند توابع فرعی را تغییر دهد تا خروجی مطلوب خود را حاصل کند اما در لایه الگوریتم صرفاً از توابع اصلی استفاده می شود. زیرا هدف هم که استاندارد سازی کد می باشد با این موضوع سازگار است.

جدول ۴-۲، توابع اصلی را نشان می دهد و جدول ۴-۳ نیز توابع کمکی را نشان می دهد. همچنین لازم به ذکر است که **کد تست استاندارد** این پروژه در به صورت زیر است:

```

1 import gym, gym_prescan
2
3 env_dict = {
4     'id':      'prescan-without-matlabengine-v0',
5     'host':    '172.21.217.140',
6     'verbose': True,
7     'nget':    152
8 }
9 env = gym.make(**env_dict)
10 for i_episode in range(20):
11     observation = env.reset()
12     for t in range(100):
13         env.render()
14         print(observation)
15         action = env.action_space.sample()
16         observation, reward, done, info = env.step(action)
17         if done:
18             print("Episode finished after {} timesteps".format(t+1))
19             break
20 env.close()

```

یادداشت ۴-۳-۲. این کد صرفاً صحت عملکرد و نحوه استفاده از رابط برنامه‌نویسی برنامه^۸ را نشان می‌دهد و شامل هیچ گونه الگوریتمی نمی‌باشد.

۴-۳-۱ معرفی برخی توابع رابط برنامه‌نویسی برنامه

بررسی تابع `__init__`:

این تابع سه وظیفه مهم دارد.

(آ) بخشی از وظایف آن، وظایف مشخص آن در کد پایتون است.

(ب) ست کردن برخی پارامترهای مهم که در جدول ۴-۱ نیز آمده‌اند.

(ج) این تابع فضای مشاهده و فضای حرکت را مشخص می‌کند.

اطلاعات فضای مشاهده و فضای حرکت در جدول ۴-۴ آمده‌است.

^۸API

نام تابع	محل استفاده	توضیحات
make	<code>--init--</code>	این تابع لایه ای را ایجاد می‌کند که هرگونه ارتباط با محیط شبیه‌سازی به آن لایه مرتبط می‌شود.
render_	<code>render</code> و <code>reset</code> و <code>step</code>	این تابع وظیفه دریافت اطلاعات از محیط شبیه‌سازی و استخراج داده‌های مفید از آن است.
send	<code>step</code>	این تابع برای فرستادن حرکت به محیط شبیه‌سازی استفاده می‌شود.
calc_reward	<code>step</code>	این تابع امتیاز را محاسبه می‌کند.
_next_observation	<code>step</code> و <code>reset</code>	این تابع حالت بعدی عامل را محاسبه می‌کند.
action_translate	<code>send</code>	این تابع مانند یک دسته‌بازی در تابع <code>send</code> حرکت را تفسیر می‌کند و دستورات کنترلی قابل اجرا برای محیط شبیه‌سازی ایجاد می‌کند.

جدول ۳-۴: راهنمای توابع کمکی رابط برنامه‌نویسی برنامه

نام متغیر	مفهوم	جنس متغیر	توضیحات
action_space	فضای حرکت	<code>spaces.Discrete(6)</code>	عدد صحیح ۶
observation_space	فضای مشاهده	<code>spaces.Box(shape=(1,38), dtype=np.float16)</code>	ماتریس ۳۸×۱ تایی

جدول ۴-۴: تعریف فضای مشاهده و فضای حرکت در پروژه

بررسی تابع `action_translation`:

این کد دقیقاً پیاده‌سازی یک دسته‌بازی^۹ می‌باشد. شکل ۴-۴ اطلاعات کامل این موضوع به همراه تفسیر آن‌ها دارد.

از آنجایی که در این پروژه فضای حرکت طبق جدول ۴-۴ مقدار عدد صحیح ۶ را دارد و این به آن معناست که ۶ حالت گسسته بین صفر تا ۵ برای حرکت وجود دارد همچنین نشان می‌دهد که جنس `action` عدد صحیح `int` است. بنابراین، می‌بایست که آن را تفسیر کرد. وظیف اصلی این تابع نیز تفسیر مقدار مختلفی است که `action` می‌تواند بگیرد، می‌باشد.

^۹Joystick


```

1 def action_translate(self,action):
2     lanewidth =
3         self.enviroment.road.laneWidth
4     self.__action_old__ = self.__action__
5     vel = self.agent['data']['Velocity']
6     offset = self.__action__[0]
7
8     if action == 0 :
9         offset = -lanewidth
10
11    if action == 1 :
12        offset = 0
13
14    if action == 2 :
15        offset = lanewidth
16
17    if action == 3 :
18        vel = action_velocity(vel,True)
19
20    if action == 4 :
21        vel = action_velocity(vel,False)
22
23    self.__action__ = [offset,vel]
24
25    return self.__action__

```



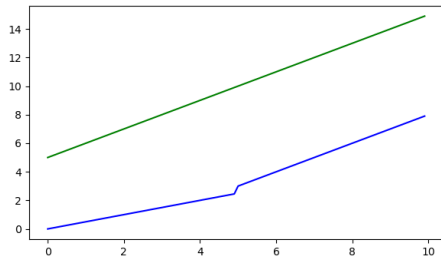
شماره	وظیفه
۰	رفتن به لاین -w
۱	رفتن به لاین 0
۲	رفتن به لاین +w
۳	زیاد کردن سرعت
۴	کم کردن سرعت
۵	بدون تغییر

شکل ۴-۴: بررسی تابع action_translate

یادداشت ۴-۳-۳. دستور کنترلی اصلی یک بردار دوتایی است (خط ۱۸ کد شکل ۴-۴) که مقدار اولی آن لاین را نشان می‌دهد و مقدار دوم آن سرعتی می‌باشد که انتظار داریم که عامل، سرعت خود را به آن برساند.

یادداشت ۴-۳-۴. اگر دقت کنید در کد مذکور دو مقدار کنونی و قدیمی تر action نگهداری شده است.

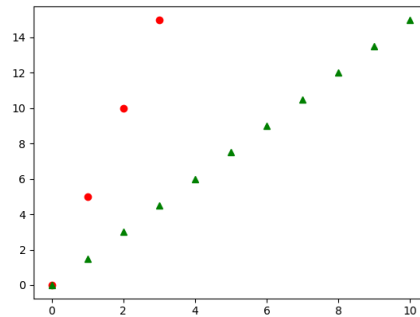
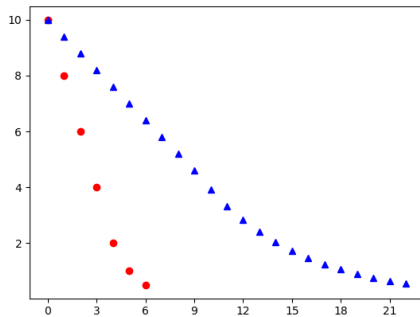
یادداشت ۴-۳-۵. همچنین دقت شود که در این کد، مقدار سرعت، همان مقدار حقیقی سرعت است که از محیط شبیه‌سازی می‌آید. و مقداری که در بردار __action__ قرار می‌گیرد مقدار کنترلی سرعت است که با یک‌دیگر تفاوت دارند.



(ب) شکل کلی نمودار افزایش سرعت

```
1 def action_velocity(vel,increase):
2     if increase:
3         return vel+5
4     else:
5         if vel < 5:
6             return vel/2
7         else:
8             return vel - 2
```

(آ) کد اصلی روند افزایشی و یا کاهش سرعت



(ج) حرکت در جهت افزایش سرعت با شروع از صفر (د) حرکت در جهت کاهش سرعت با شروع از ۱۰

شکل ۴-۵:

نکته دیگری که حایز اهمیت است این است که هنگامی که گزینه ۳ و یا ۴ انتخاب می‌شوند، سیاستی برای افزایش و کاهش سرعت اتخاذ شده است. این سیاست در قالب یک تابع در تصاویر شکل ۴-۵ ظاهر شده است. همانطور که کد ۴-۵(آ) و نمودار متناظر آن در شکل ۴-۵(ب) نشان می‌دهد، سیاست‌های مختلفی برای زیاد کردن و کم کردن سرعت اتخاذ شده است.

برای زیاد کردن سرعت، سرعت واقعی که از محیط شبیه سازی دریافت شده است با ۵ واحد جمع می‌شود. بنابراین انتظار داریم سرعت پس از سه بار افزایش به ۱۵ برسد (شکل ۴-۵(ج)، نمودار قرمز) اما این اتفاق نمی‌افتد. زیرا این افزایش سرعت، کار زمان‌بری است و نیاز به حوصله دارد که اگر حوصله و تحمل و به عبارت دیگر تاخیر را از یه حدی بالاتر ببریم عملاً در کنترل عامل به مشکل خواهیم رسید. همچنین مورد مشابه آن چه که گفته شد، در شکل ۴-۵(د) نیز برقرار است. نقاطی که رنگشان قرمز است نمودار ایدآلی مفروض خواهند بود که معادل تاخیر کم سیستم جهت اعمال سرعت نهایی است. و نمودار دیگر معادل رخدادی است که ۳۰٪ به آن عمل شده است و ۷۰٪ تحت تاثیر مقدار قبلی خواهد بود و به این صورت یک میانگین وزن‌دار گرفته شده است.

مشاهده می‌شود که مسیری که به واقعیت نزدیک‌تر است آرام‌تر از مسیر مدنظر است. همچنین

تفاوت تغییر روند کاهشی سرعت های کمتر از ۵ واحد، این است که هیچ گاه منفی نشود ولی به صورت نمایی کاهش یابد و نزدیک صفر شود.

تفاوت دیگر آن است که به هنگام افزایش مقدار ۵ واحد به سرعت افزوده می شود و در هنگام کاهش مقدار دو واحد (برای سرعت های بالای ۵ واحد) از آن کسر می شود. این تفاوت در مقایسه فاصله نقاط بین دو نمودار شکل های ۴-۵ (ج) و ۴-۵ (د) نیز ظاهر شده است. علت اصلی این تفاوت در بررسی calc_reward خود را نشان می دهد. اما پیشتر در نظر داشته باشید که یک سیاست جهت دار و تشویقی جهت قرار گرفتن سریع تر در مسیر درست می باشد.

بررسی تابع step :

کد این تابع را پیش تر بررسی شد. جایگاه این تابع داخل **کد تست استاندارد** در داخل یک حلقه است هنگامی که حلقه تمام شود به اصلاح یک اپیزود تمام شده است. دلیل تمام شدن حلقه یا رسیدن به سقف تعداد مرحله در هر اپیزود است و یا یک شدن مقدار done. این مقدار یکی از خروجی هایی است که این متغیر می تواند اختیار کند.

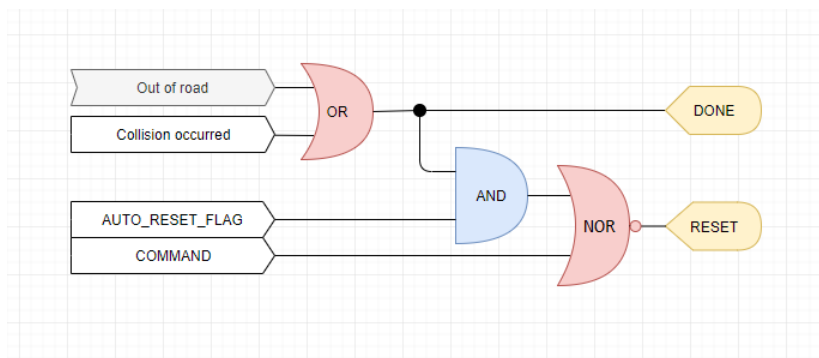
این تابع مقدار حرکت را می گیرد. سپس آن را برای محیط شبیه سازی می فرستد. در کد زیر این کار توسط تابع send انجام می پذیرد. در تابع send ابتدا با استفاده از تابع action_translate به صورت دستور کنترلی در خواهد آمد و پس از این تبدیل برای محیط شبیه سازی ارسال می شود. پس از مقداری تاخیر، متغیرهای observation و reward که به ترتیب مشاهده و امتیاز می باشند، محاسبه می شود. در کنار این دو خروجی، خروجی های دیگری وجود دارند. done که از محیط شبیه سازی می آید و نشان می دهد که اپیزود تمام شده است. متغیر info متغیری است که اطلاعات اضافه ای را خروجی می دهد که در دیباگ کردن مفید خواهد بود. این اطلاعات اضافی در این پروژه، اطلاعات برخورد و اطلاعات موقعیت عامل انتخاب شده اند.

یادداشت ۴-۳-۶. نحوه محاسبه done وظیفه محیط شبیه سازی می باشد. منطق محاسبه آن در شکل ۴-۶ آمده است. بنابراین اطلاعات دو حالت وجود دارد که موجب تمام شدن یک اپیزود و به طور معادل یک شدن done می شود:

(آ) خارج شدن از محدوده جاده ^{۱۰}

(ب) تصادف کردن با ماشین دیگر

^{۱۰} منظور از محدوده جاده، محدوده ای است که در شکل ۴-۳ (ب) مشخص شده است.



شکل ۴-۶: منطق محاسبه done در محیط شبیه‌سازی

یادداشت ۴-۳-۷. مدار منطقی شکل ۴-۶ علاوه بر محاسبه متغیر done، در مورد نحوه ریست شدن محیط نیز تصمیم‌گیری می‌کند که مرتبط به بخش فنی است و در این لایه بررسی نمی‌شود.

کد این تابع در زیر آمده است. دو تابع `_next_observation` و `calc_reward` به ترتیب مشاهده و امتیاز را محاسبه می‌کنند که به صورت جداگانه بررسی خواهند شد.

```

1 def step(self, action):
2     self.send(action)
3     # ----- BEGIN DELAY -----
4     if self.delay > 0 :
5         sleep(self.delay)
6     for _ in range(self.nget):
7         self.render_()
8         if self.done:
9             break
10    # ----- END DELAY -----
11    observation = self._next_observation()
12    reward = self.calc_reward()
13    done = self.done
14    info = {'Collision':self.collission
            , 'Position':self.agent['data']['Position']}
```

۴-۳-۲ بررسی تابع `_next_observation` :

این تابع مشخص می‌کند که چه چیزی به عنوان مشاهده اعلام شود. همان‌طور که در جدول ۴-۴ آمده است، خروجی نهایی این تابع دارای ابعاد 1×38 می‌باشد. شکل ۴-۷ این روند را به‌طور کامل نشان می‌دهد.

```

1 def _next_observation(self):
2     self.render_()
3     obs = np.zeros((1,36),dtype=np.float)
4     car = self.agent
5     theta = car['Sensors'][0]['data']['theta']
6     Range = car['Sensors'][0]['data']['Range']
7
8     for i in range(len(theta)):
9         t = int((theta[i] + 180 )/10)
10        r = Range[i]
11        if obs[0,t] != 0:
12            obs[0,t] > r
13            continue
14        obs[0,t] = r
15        extra = [car['data']["Velocity"],
16                car['data']["Position"]["y"]]
17        self.__obs__ = np.append(obs, extra)
18        return self.__obs__

```

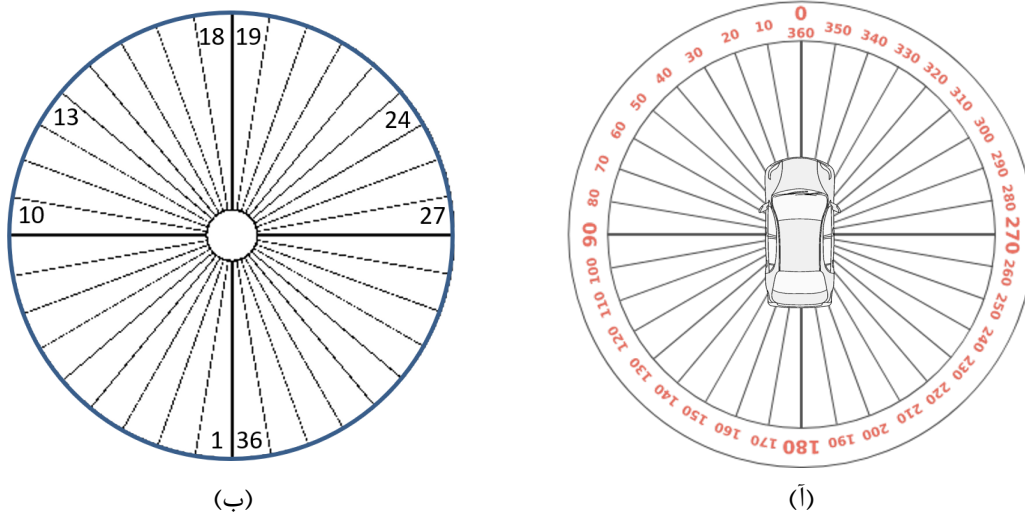
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
	20
	21
	22
	23
	24
	25
	26
	27
	28
	29
	30
	31
	32
	33
	34
	35
v	36
y	37

شکل ۴-۷: نحوه تعریف مشاهده

بنابراین مشاهده یک بردار ۳۸ تایی است که ۳۶ داده اول آن بر اساس زاویه و اندازه محاسبه شده است و ۲ داده دیگر آن مقدار y و مقدار سرعت (v) می‌باشد.

بر روی ماشین یک سنسور وجود دارد که دو بردار 10° تایی شامل فاصله و زاویه از 10° ماشین کنار خود را می‌دهد. به این ترتیب برای تعریف مشاهده فضای اطراف ماشین به ۳۶ قسمت افراز شد به طوری که هر افراز آن یک محدوده به زاویه 10° درجه را شامل می‌شود. شکل ۴-۸ طرز تعریف و مقدار دهی این بردار ۳۶ تایی را نشان می‌دهد. اگر در آن افراز ماشینی قرار نگرفت مقدار آن صفر است و در غیر این صورت فاصله آن با نزدیک‌ترین ماشین خواهد بود.

یادداشت ۴-۳-۸. از آنجایی که مقدار زاویه در بازه $\theta \in [-180^\circ, 180^\circ)$ قرار دارد، برای از بین بردن بازه منفی کل زوایا ابتدا با 180° جمع شد و سپس افراز صورت گرفت. روش دیگر می‌توانست با استفاده از هم‌نهشتی به پیمانه 360° باشد که به هنگام آزمایش روش اول کمک می‌کرد که بهتر یاد بگیرد.



شکل ۴-۸:

بررسی تابع `calc_reward`:

این تابع امتیازهای رابط برنامه‌نویسی برنامه را تنظیم می‌کند. بنابراین مهم‌ترین بخش‌های آن محسوب خواهند شد. کد این تابع در زیر آمده است.

```

1 def calc_reward(self):
2     lanewidth = self.enviroment.road.laneWidth
3     vel_sim = self.agent['data']['Velocity']
4     vel_cmd = self.__action__[1]
5     Vel = 0.8 * vel_sim + 0.2 * vel_cmd
6
7     Longitudinal_reward = reward_velocity(Vel,28) *1.5
8     Collision_reward = -25 if self.collision['Occurred'] else 0
9     Violation_reward = -0.75 * (np.abs(self.__action__[0] -
10         self.__action_old__[0])/lanewidth)
11     Nearby_reward = nearby_reward_linear(self.__obs__[12],-2.5,1.5) +\
12         nearby_reward_linear(self.__obs__[23],-2.5,1.5)
13     reward_T = Longitudinal_reward + Collision_reward +\
14         Violation_reward + Nearby_reward
15     return reward_T

```

همان‌طور که کد نیز نشان می‌دهد، امتیاز نهایی به صورت مجموع ۴ امتیاز دیگر می‌باشد. هر یک از

این امتیازها مربوط به یک بخش خاص است. بنابراین،

$$\mathcal{R}_T = \mathcal{R}_{\text{Collision}} + \mathcal{R}_{\text{Violation}} + \mathcal{R}_{\text{Longitudinal}} + \mathcal{R}_{\text{Nearby}}$$

$\mathcal{R}_{\text{Collision}}$ در صورتی که تصادف رخ دهد، مقدار آن ۲۵- و در این صورت صفر خواهد بود. بنابراین هنگام تصادف علاوه بر تمام شدن اپیزود یک امتیاز منفی با اندازه زیاد دریافت می کند.

$\mathcal{R}_{\text{Violation}}$ تعریف شد تا عامل متوجه شود که تغییر لاین هزینه خواهد داشت. مقدار این هزینه ۷۵- برابر مقدار تغییر لاین است. یعنی اگر یک واحد لاین خود را تغییر دهد، مقدار آن ۷۵- خواهد شد و اگر دو واحد لاین عوض کند، مقدار آن ۱۵- خواهد شد.

یادداشت ۴-۳-۹. اندازه مقدار $\mathcal{R}_{\text{Violation}}$ به ازای یک واحد تغییر لاین، نصف مقدار بیشینه $\mathcal{R}_{\text{Longitudinal}}$ انتخاب شده است.

یادداشت ۴-۳-۱۰. تنها امتیاز مثبت فقط $\mathcal{R}_{\text{Longitudinal}}$ با مقدار بیشینه ۱۵ می باشد.

برای محاسبه $\mathcal{R}_{\text{Longitudinal}}$ از یک تابع به نام `reward_velocity` با ویژگی های مشخص استفاده شده است. این تابع در ورودی اول خود، ورودی تابع و در ورودی دوم خود پارامتر تابع را دریافت می کند که این پارامتر مقدار ثابتی دارد.

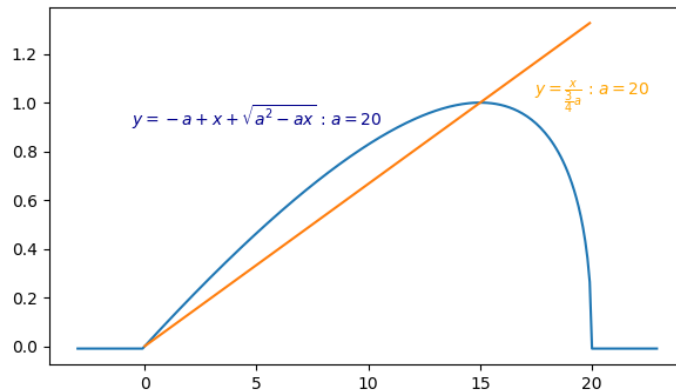
تابع مذکور، از لحاظ مقدار بیشینه بسیار خوش تعریف است و دارای ضابطه زیر است.

$$F^{\mathcal{R}_v}|_a = \begin{cases} -a + x + \sqrt{a^2 - ax} & : x \in [0, a] \\ -0.1 & : \text{سایر نقاط} \end{cases}$$

این بیشینه مقدار خود را در نقطه ۷۵a با مقدار بیشینه ۲۵a اختیار می کند. اگر این تابع را بر مقدار بیشینه اش تقسیم کنیم، نرمال خواهد شد. بنابراین:

$$F_n^{\mathcal{R}_v}|_a = \frac{F^{\mathcal{R}_v}|_a}{\max\{F^{\mathcal{R}_v}|_a\}} = \begin{cases} \frac{1}{a}(-a + x + \sqrt{a^2 - ax}) & : x \in [0, a] \\ -0.1 & : \text{سایر نقاط} \end{cases}$$

تابع `reward_velocity` به عنوان ورودی سوم تصمیم می گیرد که خروجی نرمال شده باشد یا نه. اگر `normal=True` باشد خروجی این تابع برابر خروجی تابع $F_n^{\mathcal{R}_v}|_a$ همین تابع خواهد بود و اگر



شکل ۴-۹: نمودار تابع محاسبه امتیاز سرعت نرمال شده

این مقدار False باشد، خروجی برابر خروجی تابع $F^{R_v}|_a$ خواهد بود. به صورت پیش فرض این تابع خروجی نرمال شده خواهد داشت.

نمودار این تابع در ۴-۹ آمده است که با تابع خطی که از مقدار بیشینه آن عبور می کند مقایسه شده است.

آ) تا قبل از مقدار بیشینه مقدار این دو تابع نزدیک یک دیگر می باشد و شیب آن ها نیز به هم نزدیک است.

ب) این تابع پس از مقدار رسیدن به مقدار بیشینه خود، به صورت نزولی افت می کند.

ج) اگر از نقطه ای که مقدار بیشینه در آن رخ می دهد، به سمت مثبت حرکت کنیم سریع تر از هنگامی که به سمت منفی حرکت می کند.

د) دیگر ویژگی آن این است که در حدود مقدار بیشینه خود، تقریباً هموار است که این هموار بودن آن موجب آن خواهد شد که عامل بتواند در حوالی سرعت بیشینه خود بدون تفاوت زیادی در مقدار امتیاز تصمیم بگیرد که سرعت را در صورت لزوم کم و یا زیاد کند.

نکته ۴-۳-۱۱ (خیلی مهم). چیزی که به عنوان متغیر مستقل به تابع ارسال می شود سرعت است اما با سرعت واقعی متفاوت است. در حقیقت میانگین وزن داری از سرعت واقعی و سرعت دستوری می باشد. منظور از سرعت دستوری همان سرعتی است که با استفاده از دستورات کنترلی برای محیط شبیه سازی فرستاده می شود.

$$V = 0.8V_{\text{sim}} + 0.2V_{\text{cmd}}$$

علت این تفاوت نیز به سیاست‌های تشویقی مربوط می‌شود. فرض کنید عامل تصمیم می‌گیرد سرعت را افزایش دهد. با این تصمیم مقدار سرعت ۵ واحد افزایش می‌یابد. ولی سرعت بعد از گذشت از یک مرحله به مقدار کمی سرعت آن افزایش یافته است از این رو امتیاز کمی می‌گیرد. با این میانگین‌گیری در حقیقت برای تصمیم و نیت خوب عامل مقداری امتیاز در نظر گرفته می‌شود تا مقدار امتیاز بیشتری را بگیرد.

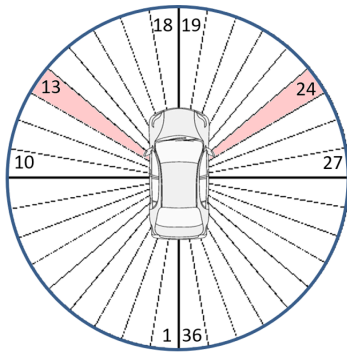
در صورتی که عامل تصمیم بگیرد که سرعت خود را کم کند، بابت این تصمیم که نامطلوب است جریمه می‌شود. مقدار این جریمه کمتر از مقدار تشویق است. این موضوع در نزدیکی‌های سرعت بیشینه کمک کننده خواهد بود. زیرا بالا بردن سرعت در آن حوالی مطلوب نیست بنابراین در صورتی که نیت کند که سرعت زیاد شود، این تصمیم عامل همراه با افت امتیاز بیشتری (نسبت به حالتی که نیت در نظر گرفته نشده است) خواهد بود. بنابراین این تفاوت‌ها مطلوب و سازنده خواهد بود.

آخرین امتیاز، $\mathcal{R}_{\text{Nearby}}$ می‌باشد. بخاطر مشکلات شبیه‌سازی در اثر تغییر لاین، این تغییر به تمامی اعمال نمی‌شود و ممکن است عامل در فاصله بسیار نزدیک از یک ماشین دیگر رانندگی کند به گونه‌ای که تصادف رخ ندهد. هدف از ایجاد این امتیاز نیز ایجاد یک حس خطر است. مسلماً باید علامت آن منفی باشد.

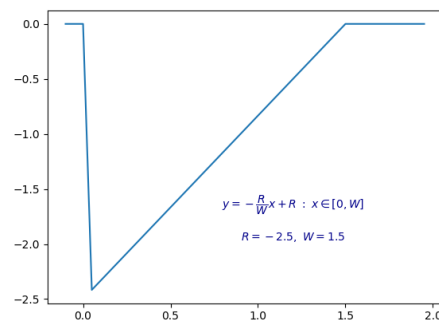
برای این منظور به نوعی دو سنسور مجازی در کناره‌های عامل کار گذاشته شد. در صورتی که ماشین دیگر در فاصله ۱/۵ متری از عامل در جهت‌های داده شده در شکل ۴-۱۰ (ب) قرار داشته‌باشد، مقدار آن صفر خواهد بود و در صورتی که فاصله از این مقدار کمتر شود به صورت خطی تا ۲/۵- امتیاز منفی دریافت می‌کند.

نمودار تابع در شکل ۴-۱۰ (ب) آمده است و ضابطه آن به صورت زیر می‌باشد:

$$F^{\mathcal{R}_{\text{Nearby}}}|_{R,W} = \begin{cases} -\frac{R}{W}x + R & : x \in [0, W] \\ 0 & : \text{سایر نقاط} \end{cases}$$



(ب) محل قرار گیری سنسورهای مجازی بر روی عامل



(آ) تابع امتیاز در زمان نزدیکی

شکل ۴-۱۰: تابع امتیاز نزدیکی و محل قرار گیری این تابع در فضای مشاهده

فصل پنجم

فنی

سلام

فصل ششم

شبیه سازی و نتایج

در فصل‌های گذشته در خصوص ابزار هایی که در این پروژه استفاده شده‌اند، صحبت شد و توضیح مفصلی بر چستی آن ابزار ها و ضرورت استفاده از آن‌ها داده شد. اما سوالات بی جوابی نیز ماند که در این فصل به آن‌ها خواهیم پرداخت.

یکی از آن سوالات نحوه راه‌اندازی کد پروژه می‌باشد و سوال دیگر نتیجه حاصل از شبیه سازی نهایی چگونه است می‌باشد.

۱-۶ راه‌اندازی

این کد در گیت هاب در دو مخزن^۱ موجود می‌باشد. ابتدا پیش‌نیاز های لازم را که در بخش ۳ توضیح داده شدند، را نصب کنید. پیشنهاد می‌شود که تمامی نسخه‌های لازم توضیح داده شده در بخش ۳ را در سیستم عامل لینوکس استفاده کنید.

اگر به توصیه استفاده از لینوکس عمل نکرده باشید، می‌توانید با استفاده شبکه که کد در اختیارتان قرار می‌دهد کد را روی دو کامپیوتر ران کنید به طوری که در یک کامپیوتر ویندوز و نرم افزار های گفته شده نصب باشد و روی دیگری لینوکس و پایتون و پیش‌نیاز های پایتون که در بخش ۳-۴ بررسی شدند نصب باشد.

یادداشت ۱-۱-۶. بهتر است این دو کامپیوتر در یک شبکه داخلی به یک دیگر متصل شده باشند.

حال وارد کامپیوتری شوید که ویندوز بر روی آن نصب است. این کامپیوتر قرار است نقش محیط را برای ما ایجاد کند.

یادداشت ۲-۱-۶. کد های پایتون صرفاً بر روی کامپیوتری که لینوکس دارد اجرا کنید.

که دستور زیر را در ترمینال^۲ خود وارد کنید.

```
1 git clone https://github.com/MohammadRaziei/gym-Prescan.git
2 pip install -e gym-Prescan
```

خط دوم این کد، اختیاری می‌باشد و صرفاً کار را ساده می‌کند. همچنین می‌توان آن را به صورت زیر نیز نوشت:


```
1 pip install git+https://github.com/MohammadRaziei/gym-Prescan
```

¹Repository

²Terminal

سپس وارد مسیر زیر شوید.

`gym-Prescan/gym_prescan/envs/PreScan`

سپس با استفاده از آیکون ^۳ کلیک کنید. در این صورت بر روی نوار Toolbar این آیکون نیز ظاهر می شود. با فشردن آن، پنجره شکل ۳-۳ باز می شود. بر روی Matlab کلیک کنید تا محیط متلب باز شود. اجرای فایل `startup.m` برای هنگامی که از کد پایتون بر روی همان سیستم استفاده نمی کنید، اختیاری است.

فایل سیمولینک را باز کرده و

پس حال در سیستم لینکوس خود میتوانید بسته زیر را دانلود کنید.

```
1 git clone https://github.com/MohammadRaziei/gym-Prescan-minimal.git
2 cd gym-Prescan-minimal
```

در این پوشه تعدادی از الگوریتم های معروفی که در حوزه یادگیری تقویتی عمیق نوشته شده است، قرار دارد. در بین این الگوریتم ها دو الگوریتم DQN و A2C نسبت به بقیه بهتر جواب داده اند. این الگوریتم ها در بخش؟؟ و در [۵] توضیح کامل داده شده اند.

^۳ این آیکون پس از نصب نرم افزار پری اسکن بر روی دستکتاپ تشکیل می شود.

منابع و مراجع

- [1] D. Silver, “UCL course on RL.” <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>, 2015.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.
- [3] G. Hayes, “How to install openai gym in a windows environment.” <https://medium.com/p/338969e24d30>, 2018.
- [4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.
- [5] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Stable baselines.” <https://github.com/hill-a/stable-baselines>, 2018.

نمایه

مخزن، ۲۷	حرکت، ۲۱، ۸۶
امتیاز، ۲۲، ۲۰، ۷۵	عامل، ۲۱، ۲۰، ۱۷، ۹۶
فرضیه امتیازها، ۵	حالت عامل، ۲۰، ۸
یادگیری تقویتی، ۶۴، ۱۶، ۲۰، ۲۲	الگوریتم، ۱۷، ۱۶
یادگیری شبه ناظر، ۴	ماشین خودران، ۲۰
حالت، ۲۲، ۸۶	نقطه تراز، ۱۶
مرحله، ۲۱، ۶	خوشه بندی، ۴
ناظر، ۵	هزینه، ۵
یادگیری با ناظر، ۴	یادگیری تقویتی عمیق، ۲۸، ۱۷
ترمینال، ۲۷	محیط، ۲۷، ۲۰، ۱۷، ۱۶، ۹، ۷، ۶
تست، ۲۳، ۲۲	حالت محیط، ۲۰، ۸، ۷
آموزش، ۲۲	اپیزود، ۲۱
یادگیری بدون ناظر، ۴	فیدبک، ۵
	تاریخچه، ۸، ۷
	حالت اطلاعاتی، ۹
	مارکوف، ۸
	حالت مارکوف، ۹، ۸
	متلب، ۱۷
	یادگیری ماشین، ۴
	مشاهده، ۷، ۶
	مشاهده پذیری، ۲۰، ۸
	پری اسکن، ۲۸

فهرست اختصارات

A

A2C Synchronous Actor Critics

D

DQN Deep Q-Learning Network

R

RL Reinforcement Learning

واژه‌نامه انگلیسی به فارسی

E

Environment محیط
Environmnet State حالت محیط
Episode اپیزود

F

Feedback بازخورد
Full observability ... مشاهده‌پذیری کامل

H

History تاریخچه

I

Information State حالت اطلاعاتی

J

Joystick دسته‌بازی

M

Machine Learning یادگیری ماشین

A

Action حرکت
Action Space فضای حرکت
Agent عامل
Agent State حالت عامل
Algorithm الگوریتم
Anaconda نرم‌افزار آناکوندا
API رابط برنامه‌نویسی برنامه
Autonomous Vehicle ماشین خودران

B

Bechmark نقطه تراز

C

Clustering خوشه بندی
Cost هزینه

D

Deep یادگیری تقویتی عمیق
Reinforcement Learning
Dynamic Data داده‌های پویا

Step مرحله
 Supervised Learning یادگیری با ناظر
 Supervisor ناظر

T

Terminal ترمینال
 Test تست
 Timeout سقف زمانی
 Train آموزش

U

Unsupervised Learning یادگیری بدون ناظر

Markov مارکوف
 Markov Decision Process روند تصمیم‌گیری مارکوف
 Markov State حالت مارکوف
 Matlab نرم‌افزار متلب
 Matlab Engine موتور متلب

O

Observability مشاهده‌پذیری
 Observation مشاهده
 Observation Space فضای مشاهده

P

Partial observability مشاهده‌پذیری جزئی
 PreScan نرم‌افزار پری‌اسکن

R

Reinforcement Learning یادگیری تقویتی
 Repository مخزن (گیت‌هاب)
 Reward امتیاز
 Reward Hypothesis فرضیه امتیازها

S

Semi-Supervised Learning یادگیری شبه ناظر
 Simulink سیمولینک
 State حالت

واژه‌نامه فارسی به انگلیسی

خ	ا
Clustering خوشه بندی	Train آموزش
	Episode اپیزود
	Algorithm الگوریتم
د	Reward امتیاز
Dynamic Data داده‌های پویا	
Joystick دسته‌بازی	ب
	Feedback بازخورد
ر	
API رابط برنامه‌نویسی برنامه	ت
	History تاریخچه
س	Terminal ترمینال
Timeout سقف زمانی	Test تست
Simulink سیمولینک	
	ح
ع	State حالت
Agent عامل	Information State حالت اطلاعاتی
	Agent State حالت عامل
	Markov State حالت مارکوف
ف	Environmnet State حالت محیط
Reward Hypothesis فرضیه امتیازها	Action حرکت
Action Space فضای حرکت	

یادگیری تقویتی . Reinforcement Learning
 یادگیری تقویتی عمیق Deep
 Reinforcement Learning
 یادگیری شبه ناظر Semi-Supervised
 Learning
 یادگیری ماشین Machine Learning

فضای مشاهده Observation Space

م

مارکوف Markov
 ماشین خودران Autonomous Vehicle
 محیط Environment
 مخزن (گیت‌هاب) Repository
 مرحله Step
 مشاهده Observation
 مشاهده‌پذیری Observability
 مشاهده‌پذیری جزئی Partial observability
 مشاهده‌پذیری کامل Full observability
 موتور متلب Matlab Engine

ن

ناظر Supervisor
 نرم‌افزار آناکوندا Anaconda
 نرم‌افزار پری‌اسکن PreScan
 نرم‌افزار متلب Matlab
 نقطه تراز Bechmark

ه

هزینه Cost

ی

یادگیری با ناظر Supervised Learning
 یادگیری بدون ناظر Unsupervised Learning