به نام حق

پروژه درس هوش مصنوعی

استاد درس : دکتر احد هراتی

شوت یک ضرب

پاییز ۹۶

مقدمه

یکی از زیر شاخه‌های ربوکاپ، لیگ شبیه‌سازی ۲ بعدی است که تمرکز این لیگ بر روی نرم‌افزار، بستر مناسبی را برای پیاده‌سازی الگوریتم‌های هوش مصنوعی فراهم می‌آورد.

در این محیط شبیه‌سازی شده تلاش شده است بسیاری از اعمال و حسگرهای ربات‌های بازیکن واقعی و شرایط واقعی بازی شبیه‌سازی شود. از این رو هر بازیکن به طور مجزا از بازیکن‌های دیگر، اطلاعاتی را به صورت string و از طریق پروتکل UDP/IP برای سرور شبیه‌ساز ارسال و از آن دریافت می‌کند. اطلاعاتی که هر بازیکن دریافت می‌کند شامل اطلاعات محیط مانند فاصله بازیکن تا نقاط مختلف زمین و مکان نسبی توپ نسبت به بازیکن است که این اطلاعات همراه با خطاست. همچنین هر بازیکن اطلاعاتی مانند چگونه حرکت کردن و یا شوت کردن را برای سرور می‌فرستد.

از وظایف دیگر سرور داوری و پیش بردن زمان است. زمان به صورت گسسته و تحت عنوان cycle معرفی می‌شود. هر سایکل معادل ۱۰۰ میلی‌ثانیه است. در هر سایکل تنها یک دستور kick و یا move قابل اجراست اما دستوراتی مانند تغییر زاویه دید (turn neck) می‌تواند بارها در کنار سایر دستورات در یک سایکل اجرا شود. اگر دستورات جابه‌جایی یک شی (توپ یا بازیکن) در یک سایکل به سرور ارسال شود، شی مربوطه در ابتدای سایکل بعدی جابه‌جا خواهد شد.

نرم‌افزار سرور بازی فوتبال را شبیه‌سازی می‌کند اما آن را به تصویر نمی‌کشد. از این رو نرم‌افزار نمایشگر شبیه‌ساز فوتبال ربوکاپ (RCSS Monitor) به عنوان یک client به سرور شبیه‌ساز متصل شده و وقایع درون بازی را به صورت گرافیکی نمایش می‌دهد.

شرح پروژه

بازی شوت یک ضرب به این صورت است که از هر تیم یک بازیکن در زمین قرار می‌گیرد و هر بازیکن تلاش می‌کند توپ را درون دروازه حریف جای دهد با این محدودیت که بازیکنان در هر نوبت تنها یک‌بار مجاز به لمس توپ هستند.

لازم است تیم شما بدون شانس، به صورت قطعی و با نتیجه مقتدرانه برنده این بازی شود.

قوانین بازی:

- برای شبیه‌سازی بازی شوت‌یک ضرب، در سرور شبیه‌سازی ربوکاپ تغییراتی از نظر داوری و ارسال و دریافت پیام بین سرور و agent ها اعمال شده است که لازم است این سرور خاص‌منظوره که در پیوست آمده است را نصب کنید.
- ابعاد زمین در این سرور، ۴۰ در ۲۵ می‌باشد.
- هر تیم یک Agent بازیکن (و نه goalie) را به سرور متصل می‌کند.
- در طول اجرای بازی اگر یک Agent توپ را دو بار پیاپی kick کند (یعنی در بین ضربات او حریف ضربه‌ای نزده باشد) یک امتیاز برای تیم حریف حساب خواهد شد.
- اگر نوبت یکی از Agent ها باشد که به توپ ضربه بزند اما تا ۶ ثانیه به توپ ضربه‌ای وارد نکند، یک امتیاز برای حریف حساب می‌شود.
- سرور در هر سیکل اطلاع دقیق در رابطه با آخرین بازیکنی که توپ را لمس کرده را به هر agent می‌فرستد که برای این منظور می‌توانید از تابع accurateLastKickerSide در کلاس WorldModel استفاده کنید.
- فرض بر این است که زمین بازی با دیوار محصور شده است. بنابراین هیچوقت توپ از زمین بازی خارج نمی‌شود و گل دیواری نیز قابل قبول است.

نکات

- کدهای مربوط به سرور، مانیتور، تیم‌ها و هرآنچه که نیاز دارید را می‌توانید از طریق لینک github.com دانلود نمایید.
- در **پیوست** مراحل نصب و آماده‌سازی تیم‌ها و راهنمایی برای آشنایی بیشتر با بیس agent2d آمده است.
- زمان تحویل پروژه، پس از آخرین روز امتحانات است که زمان دقیق آن متعاقبا توسط گروه حل‌تمرین اعلام خواهد شد.
- پیاده‌سازی پروژه به صورت **انفرادی** است. تحویل غیر حضوری پروژه از طریق لینک dropbox.com و تحویل نهایی آن به صورت حضوری می‌باشد.
- در صورت داشتن ابهام و یا مشکلی در نصب و راه‌اندازی سرور و تیم‌ها می‌توانید از طریق یکی از ایمیل‌های rza.ete@gmail.com و یا Hosein.mohebbi75@gmail.com اقدام نمایید.

نصب و آماده‌سازی محیط شبیه‌سازی و تیم‌ها:

- همراه با صورت پروژه ۲ دایرکتوری با نام‌های teams و oneStepShoot پیوست شده است که به ترتیب کدهای مربوط به تیم‌ها و شبیه‌سازهای بازی (سرور و مانیتور) می‌باشد که لازم است کامپایل شوند.
- قبل از نصب با اجرای دستور زیر در ترمینال، وابستگی‌های ضروری را نصب کنید.

```
$ sudo apt-get install g++ build-essential libboost-all-dev qt4-dev-tools libaudio-dev
$ sudo apt-get install libgtk-3-dev libxt-dev bison flex
```

- سپس باری نصب سرور و مانیتور وارد دایرکتوری oneStepShoot شده و اسکریپت challengeSet را با استفاده از ترمینال اجرا کنید.
- دستور فوق تنها یکبار لازم است اجرا شود و بعد از نصب، از این پس می‌توانید با اجرای اسکریپت challengeRun در همین دایرکتوری، سرور و مانیتور را اجرا کنید.
- برای کامپایل کد تیم‌تان وارد دایرکتوری teams شده و اسکریپت compile را اجرا کنید.
- بعد از کامپایل، برای اجرای کد تیم‌تان به مسیر teams/yourTeam/src رفته و اسکریپت start.sh را اجرا کنید. توجه کنید برای اینکه تغییراتی که در کدهایتان اضافه کردید اعمال شود لازم است کدتان را هربار با استفاده از دستور make کامپایل کنید.
- برای اجرا کد تیم حریف لازم است به مسیر teams/computerTeam/src رفته و اسکریپت start.sh را اجرا کنید. توجه کنید که باینری تیم حریف(کامپیوتر) به شما داده شده است بنابراین نیازی به کامپایل کد ندارید.
- بعد از اجرای کد هر دو تیم بر روی سرور و اضافه شدن تیم‌ها به مانیتور، میتوانید با کلید ترکیبی ctrl+k بازی را شروع کنید.

راهنمای بیس Agent2D

کلاسی به نام LocomotiveAgent در اختیار شما قرار داده شده است. در این کلاس تابعی به نام execute وجود دارد. در هر سیکل، این تابع یک بار صدا زده میشود و شما باید در این تابع دستوراتی را که میخواهید، به ایجنت خود بدهید.

تابع execute به عنوان ورودی یک شی از نوع کلاس PlayerAgent میگیرد. این کلاس حاوی متدی به نام world است که یک شی از نوع کلاس WorldModel برمیگرداند. این شی بیانگر حالت کنونی جهان است و درون آن، تمامی اطلاعاتی قرار دارد که شما به آنها دسترسی دارید.

کلاس WorldModel

این کلاس شامل اطلاعاتی از شرایط حال حاضر بازی است. دقت کنید که در هر سیکل این اطلاعات تغییر میکنند و اگر شما میخواهید که به اطلاعات گذشته دسترسی داشته باشید باید خودتان آن را ذخیره کنید. در زیر به طور خلاصه به تعدادی از متدهای این کلاس اشاره میشود. لازم به ذکر است که همه متدها دارای توضیحات هستند و با خواندن آنها میتوانید متوجه عملکرد متدها شوید.

متد self

این متد شی ای از نوع کلاس SelfObject برمیگرداند. این کلاس حاوی اطلاعاتی در مورد ایجنت است.

متد ball

این متد شی ای از نوع کلاس BallObject برمیگرداند. این کلاس حاوی اطلاعاتی در مورد توپ است.

متد opponentsFromBall

این متد آرایه ای از بازیکنان حریف (vector<PlayerObject*>) برمیگرداند که به ترتیب فاصله از توپ مرتب شده اند. (در این بازی از هر تیم یک بازیکن حضور دارد بنابراین طول این آرایه هیچ وقت بزرگتر از 1 نمیشود.)

متد ourSide

خروجی این متد از نوع SideID است که یک enum است. اگر تیم ما سمت چپ زمین باشد مقدار LEFT یا 1 و اگر سمت راست زمین باشد مقدار RIGHT یا 1- برمیگرداند.

متد theirSide

خروجی این متد نیز از نوع SideID است. اگر تیم حریف سمت چپ زمین باشد مقدار LEFT یا 1 و اگر سمت راست زمین باشد مقدار RIGHT یا 1- برمیگرداند.

متد time

این متد شی ای از نوع کلاس GameTime برمیگرداند. با استفاده از متد cycle تعریف شده در این کلاس، میتوان سیکل جاری را دریافت کرد.

متد accurateLastKickerSide

این متد side تیمی که آخرین بار به توپ ضربه زده است را برمیگرداند.

کلاس AbstractPlayerObject

این کلاس حاوی اطلاعاتی در مورد ایجنت ها است.

متد pos

این متد بردار مکان ایجنت را برمیگرداند.

متد vel

این متد بردار سرعت ایجنت را برمیگرداند.

متد body

این متد زاویه بدن ایجنت را برمیگرداند.

متد distFromBall

این متد فاصله ایجنت از توپ را برمیگرداند.

متد distFromSelf

این متد فاصله یک ایجنت را از ایجنتی که در حال اجرای برنامه است، برمیگرداند.

متد inertiaPoint

این متد با دریافت یک عدد صحیح n ، محاسبه میکند که ایجنت با توجه به سرعت و مکان آن در سیکل جاری، بعد از گذشت n سیکل به طور تخمینی در چه مکانی قرار میگیرد.

متد inertiaFinalPoint

این متد محاسبه میکند که ایجنت با توجه به سرعت و مکان آن در سیکل جاری، به طور تخمینی در چه مکانی متوقف میشود.

کلاس PlayerObject و SelfObject

این کلاس ها هر دو از کلاس AbstractPlayerObject ارث برده اند بنابراین تمام متدهای بالا را دارند.

متد isKickable

این متد چک میکند که آیا توپ در فاصله ای هست که ایجنت بتواند شوت بزند یا نه.

کلاس BallObject

این کلاس حاوی اطلاعاتی در مورد توپ است.

متد pos

این متد بردار مکان توپ را برمیگرداند.

متد vel

این متد بردار سرعت توپ را برمیگرداند.

متد distFromSelf

این متد فاصله توپ را از ایجنتی که در حال اجرای برنامه است، برمیگرداند.

متد inertiaPoint

این متد با دریافت یک عدد صحیح n ، محاسبه میکند که توپ با توجه به سرعت و مکان آن در سیکل جاری، بعد از گذشت n سیکل به طور تخمینی در چه مکانی قرار میگیرد.

متد inertiaFinalPoint

این متد محاسبه میکند که توپ با توجه به سرعت و مکان آن در سیکل جاری، به طور تخمینی در چه مکانی متوقف میشود.

کلاس Vector2D

این کلاس برای کار با مختصات و بردارها نوشته شده است و حاوی متدهایی برای دریافت طول بردار، زاویه بردار، بردار یکه و همچنین متدهایی برای محاسبه فاصله دو نقطه، جمع و تفریق دو بردار، دوران دادن بردار و ... است.

کلاس ServerParam

این کلاس برای نگهداری و دسترسی به پارامترهای مربوط به سرور و بازی که مقدار ثابتی دارند استفاده میشود. این کلاس با پترن singleton نوشته شده است و برای دسترسی به آن باید از متد استاتیک i یا instance استفاده کنید.

متد pitchLength

این متد اندازه طول زمین را برمیگرداند.

متد pitchWidth

این متد اندازه عرض زمین را برمیگرداند.

متد goalWidth

این متد اندازه عرض دروازه را برمیگرداند.

متد ourTeamGoalPos

این متد مختصات دروازه خود را برمیگرداند.

متد theirTeamGoalPos

این متد مختصات دروازه حریف را برمیگرداند.

Action های قابل استفاده

در پوشه LocomotiveLib/rcsc/action نمونه هایی از action هایی که یک ایجنت میتواند انجام دهد، تعریف شده است. در این بازی با توجه به قوانین تعریف شده فقط از تعدادی از این action ها میتوانید استفاده کنید که در زیر به آنها اشاره میشود. همچنین نمونه هایی از استفاده از این action ها در متدهای doKick و doDash کلاس LocomotiveAgent آورده شده است.

Body_KickOneStep

از این action برای شوت کردن توپ میتوانید استفاده کنید. سازنده این کلاس دو پارامتر دریافت میکند.
پارامتر اول از نوع Vector2D است و مختصات مقصد را مشخص میکند.
پارامتر دوم از نوع double است و سرعت اولیه توپ را مشخص میکند.

Body_GoToPoint2010

از این action برای حرکت در زمین میتوانید استفاده کنید. سازنده این کلاس سه پارامتر دریافت میکند.
پارامتر اول از نوع Vector2D است و مختصات مقصد را مشخص میکند.
پارامتر دوم از نوع double است و آستانه قابل قبول برای رسیدن به مقصد را مشخص میکند. به عنوان مثال اگر این مقدار 1 باشد به این معنی است که اگر ایجنت در شعاع یک متری نقطه مقصد (پارامتر اول) قرار بگیرد، به مقصد رسیده است.
پارامتر سوم از نوع double است و حداکثر سرعتی را که ایجنت میتواند داشته باشد، مشخص میکند.

Body_Intercept

از این action برای تصاحب توپ میتوانید استفاده کنید. سازنده این کلاس هیچ پارامتری دریافت نمیکند.
با اجرای این action بر اساس مسیر و سرعت حرکت توپ، بهترین نقطه برای سد کردن توپ محاسبه میشود و ایجنت به آن نقطه میرود.

اضافه کردن فایل جدید به پروژه

برای اضافه کردن فایل جدید به پروژه باید مراحل زیر را طی کنید.

1. ابتدا فایل های جدید را در پوشه LocomotiveSrc/src ایجاد کنید.
2. در این پوشه فایلی به نام Makefile.am وجود دارد. آن را باز کنید و نام فایل cpp خود را در قسمت PLAYERSOURCES و نام فایل h خود را در قسمت PLAYERHEADERS اضافه کنید.

به عنوان مثال اگر بخواهیم کلاسی به نام Test اضافه کنیم باید دو فایل test.cpp و test.h بسازیم و به صورت زیر به Makefile.am اضافه کنیم.

```
PLAYERSOURCES = \
        test.cpp \
        bhv_basic_move.cpp \
        bhv_basic_offensive_kick.cpp \

PLAYERHEADERS = \
         test.h \
         bhv_basic_move.h \
         bhv_basic_offensive_kick.h \
```

3. در انتها اسکریپت configNewFiles قرار داده شده در پوشه Locomotive را با دستور configNewFiles/. اجرا کنید.

موفق باشید