دانشگاه صنعتي اميركبير
(پلی تکنیک تهران)
دانشکده ریاضی و علوم کامپیوتر

درس هوش مصنوعی و کارگاه

گزارش 3: بازی پکمن

نگارش
رامان منبری اسکویی
40013029

استاد اول
دکتر مهدی قطعی

استاد دوم
دکتر بهنام یوسفی مهر

آبان 1402

چکیده

در این پروژه سعی بر این بوده که برای بازی پکمن پیاده سازی الگوریتم minimax و کد alpha-beta pruning پیاده سازی بشود. در این پروژه برای بازی پکمن از کد سایت دانشگاه برکلی استفاده شده است. در ادامه به توضیح تابع minmax و alpha-beta pruning می پردازیم که به چه شکل این توابع پیاده سازی شده است.

واژه‌های کلیدی:
هوش مصنوعی،پکمن،minimax، alpha-beta pruning،

فهرست مطالب

صفحه

چکیده		1
فصل اول بازی پکمنبازی پکمن		4
فصل دوم توضیحات و اجرا کردن کداجرا کردن کد		6
2-1-	توضیحات کد evaluationFunction	7
2-2-	توضیحات کد minimaxagent	7
2-3-	کد کارایی	7
فصل سوم جمع‌بندي جمع‌بندي		10

فصل اول
بازی پکمن

بازی پکمن

بازی پکمن از یک سری موانع و یک سری نقاط تشکیل شده است. با خوردن هر نقطه 10 امتیاز به دست اورده و با هر حرکت یک امتیاز منفی به دست می اوردیم. پکمن توانایی پریدن از موانع را ندارد بنابراین توانایی عبور از موانع را نداریم. در بازی چند روح وجود دارند که اگر پکمن ما توسط این روح ها خورده شود پکمن بازنده است. پکمن در یک صورت برنده بازی است، خوردن تمام نقاط بدون خورده شدن توسط روح ها. روح ها در بازی پکمن به صورت رندوم حرکت می کنند و ما هیچ گونه ایده از حرکت انها نداریم.

در این پروژه سعی بر این شده است که به جای بازیکن واقعی از هوش مصنوعی که به صورت minimax و alpha-beta pruning عمل می کند استفاده شود.

فصل دوم
توضیحات و اجرا کردن کد

اجرا کردن کد

برای اجرا کردن پوشه ای که کد ها در ان قرار دارند رو باز کرده و در قسمت ادرس پوشه کلمه cmd رو نوشته و دکمه اینتر را می زنیم. با این حرکت cmd باز شده و می توانیم دستور اجرا کد را بزنیم.

- python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4 این دستور برای اجرا بازی به صورت minimax است که می توانید با عوض کردن نقشه بازی (که در اینجا minimaxClassic است) پکمن را در محیط های مختلف تست کنید. برای نقشه های مختلف می توانید به پوشه layouts مراجعه کرده و اسم هر نقشه ای که می خواهید پکمن در آن شروع به بازی کند در دستور اجرای کد بنویسد. برای اینکه در عمق های مختلف الگوریتم minimax اجرا شود می توانید depth ان را تغییر دهید. پیشنهاد depth 4 است به دلیل اینکه هم با سرعت خوبی شروع به بازی کردن می کند و مدت زمان برای تصمیم گیری عدد معقولی است ولی با بالا بردن depth مشکلی در اجرا کد پیش نمی آید.(والا depth 10 زدم هنگ کرد نکشید لپتاپ).
- python pacman.py -p AlphaBetaAgent -a depth=4 -l smallClassic این دستور برای اجرا کد به صورت alpha-beta pruning است که می توانید مانند قبل مپ بازی و عمق را می توانید عوض کنید.

2-1- توضیحات کد **evaluationFunction**

در این قسمت سعی بر این شده است که بتوان تابع e-utility پیاده سازی شود. حال به بررسی این موارد می پردازیم. در اولین گام تلاش بر این بوده نزدیک ترین فاصله منهتی از موقعیت پکمن تا نزدیک ترین غذا پیدا شود با این حرکت سعی بر این می شود که کمترین هزینه برای رسیدن به غذا صرف شود.در مرحله بعد نزدیک ترین کپسول را پیدا می کنیم با روش فاصله منهتن. در گام بعدی سعی می کنیم نزدیک ترین روح به خود را پیدا کنیم که اگر به شدت نزدیک بود بتوانیم از دست روح فرار کنیم و شکست نخوریم. در مرحله بعد اگر تعداد گوست ها دور بزرگ تر از صفر بود و فاصله انها کمتر 2(یک خانه نزدیک ما بود) به این معناست است که ما نمی توانیم بازی رو ببریم باید در سریع ترین حالت ممکن ببازیم تا بتوانیم ماکسیمم امتیاز را به دست بیاریم. ولی اگر به این صورت نبود امتیاز هر یک از راه های گفته شده را به value اضافه می کنیم و در نهایت مقدار value را برمیگردانیم.

2-2- توضیحات کد **minimaxagent**

در این قسمت سعی بر این شده است که تابع minimax در جزوه به آن اشاره شده است پیاده سازی شود. در ابتدا بهترین حرکت که مساوی بیشترین ماکسیسمم امتیاز است که در آن موقعیت است استفاده میشود. حال تابع getMinValue که به این صورت کار می کند اگر نتواند کاری کند در همان موقعیت می ایستد. اگر بر روی agent صفر بودیم یکی از عمق کم کرده و تابع getMaxvalue را صدا زده در غیر این صورت تابع getMinvalue را صدا زده. مینیمم مقدار را به عنوان بهترین مقدار(چون تابع getMinValue است) انتخاب می کنیم. حرکت و بهترین مقدار را خروجی می دهیم.

تابع getMaxValue هم مانند جزوه و برعکس تابع getMinValue تعریف میکنیم.

در این پروژه سعی بر این بوده است تمامی تعاریف و کد ها بر اساس جزوه زده شود حتی الگوریتم alpha-beta pruning

2-3- کارایی کد

از انجایی که برای حرکات روح ها رندوم است لزوما با اجرای یکبار کد به نتیجه ای که می رسیم نمی توانیم اکتافا کنیم. اول از همه درباره عمق صحبت کنیم. کد برای هر عمقی به صورت دقیق کار می کند و مشکلی از لحاظ اجرا نشدن کد وجود ندارد ولی برای عمق های بالا(مثل 10 و 11) به مشکل هنگ کردن لپتاپ و سیستم برخورد کردم. عمق ای من بیشتر برای اجرای کدم استفاده کردم عمق 3 و 4 بوده است. چون مدت زمان تصمیم گیری برای پکمن هم عدد معقولی است و هم سیستم هنگ نمی کند. برای مثلا برای عمق 6 تقریبا برای هر حرکت پکمن نزدیک به 6 ثانیه طول می کشید تا محاسبه شود.حال به بررسی الگوریتم minimax در نقشه های مختلف می پردازیم.

- نقشه minimaxClassic: در این نقشه تا زمانی که روح ها پکمن ما رو مقایسه نکند پکمن از دست روح ها فرار کرده تا بتواند در زمان درست اخرین نقطه را خورده و برنده بازی شود.
- نقشه openClassic: در این نقشه پکمن شروع به خوردن نقطه های نزدیک خورده کرده ولی به دلیل اینکه بعد از خوردن نقطه های نزدیک خود نزدیک ترین نقطه به او فاصله زیادی دارد تا زمانی که روح

به روح به دنبال پکمن نیتفد تا پکمن از دست روح فرار کند به سمت دیگری نمی رودو البته با افزایش عمق می توانیم این مشکل را برطرف کنیم.

- نقشه trappedClassic: در این نقشه به دلیل اینکه در دو طرف پکمن روح وجود دارد پکمن سعی در خودکشی تا کمترین امتیاز را از دست بدهد.البته در یک صورت که هر دو روح در جهت نفع پکمن حرکت کنن پکمن شروع به خوردن نقاط می کند.(بعد از حدود 25 بار اجرا کردن همچین اتفاقی افتاد)

در تمامی نقشه ها پکمن سعی در به دست اوردن امتیاز ماکسیسمم ممکنه و در صورتی که روحی به ان نزدیک می شد سعی در فرار و اگر توسط روح ها محاصره می شد سعی بر خودکشی می کرد.

فصل سوم
جمع‌بندي

جمع‌بندي

در این پروژه سعی بر این بوده که توابع minimax و alphabeta-purning با توجه به توضیحات درون جزوه پیاده سازی شود.

در این پروژه اول بازی پکمن پیاده سازی شده است و بعد تابع های util و minimax و alphabeta-purning که بتواند جای پکمن بازی کند.مسلما تمامی تابع ها به طور کامل به صورت بهینه کار نمی کنه و می توان انها را بهبود بخشید.

.

منابع

[1] https://inst.eecs.berkeley.edu/~cs188/fa22/projects/proj2/