گزارش فاز سوم پروژه: بازی Connect four

مبانی و کاربردهای هوش مصنوعی

دکتر حسین کارشناس

آدرینا ابراهیمی ۹۹۳۶۲۳۰۰۲

کیان مجلسی ۹۹۳۶۱۳۰۵۱

خرداد ۱۴۰۲

شرح مسئله

در این مرحله، از محیط بازی Connect four از کتاب‌خانه PettingZoo استفاده شده است. در این محیط، صفحه بازی به صورت یک صفحه ۶ در ۷ بوده و هر بازیکنی که بتواند چهار مهره خود را به صورت پشت سر هم در یکی از حالت‌های عمودی، افقی یا قطری بگذارد برنده می‌شود. در این قسمت به طراحی عاملی می‌پردازیم که بتواند پس از انتخاب کنش توسط انسان بتواند بهترین بازی خودش را انجام دهد.

نام و علت انتخاب الگوریتم

از الگوریتم Minimax با استفاده از هرس آلفا و بتا استفاده شده است. عملکرد این الگوریتم به این صورت است که عامل در نظر می‌گیرد بهترین کنش از سمت ما انتخاب شده که منجر به بیشترین ضرر به آن می‌شود (کاهش حداکثری پاداش عامل). از این رو، عامل نیز بهترین کنش را انتخاب کرده که منجر به افزایش حداکثری پاداش خودش می‌شود. به عبارتی، فرض عامل فرض می‌کند ما بهترین بازی را انجام می‌دهیم. از این رو، او نیز بهترین بازی خود را انجام می‌دهد.

گزارش کار الگوریتم

تابع heuristic:

این تابع با دریافت یک حالت از بازی و بازیکن، با توجه به چینش مهره‌های بازیکن و حریفش به صورت عمودی، افقی و قطری یک امتیاز برای آن محیط برمی‌گرداند.

تابع winning_condition:

این تابع با دریافت یک حالت از بازی و بازیکن شرط برنده شدن را برای آن بازیکن بررسی می‌کند.

تابع generate_map:

این تابع با دریافت بر اساس مشاهده دریافتی از محیط یک نقشه جدید از محیط را برای هر بازیکن تولید کرده و با ترکیب آن‌ها یک نقشه کلی از محیط را به صورتی ایجاد می‌کند که برای عامل هوشمند

در خانه‌ای که مهره وجود دارد عدد ۱ و برای بازیکن انسان در خانه‌ای که مهره قرار دارد عدد ۲ را قرار داده و نقشه محیط را برمی‌گرداند.

تابع is_terminal_node:

در این تابع بررسی می‌شود آیا نقشه دریافتی از محیط نشان‌دهنده برنده بودن عامل هوشمند یا بازیکن انسان یا برقرار بودن شرط مساوی است یا خیر.

تابع get_valid_columns:

این تابع با دریافت یک نقشه از محیط، ستون‌هایی را که امکان قرار دادن مهره در آن وجود دارد را به صورت یک لیست برمی‌گرداند.

تابع get_next_open_row:

این تابع با دریافت یک نقشه از محیط و یک ستون، اولین سطری که مهره می‌تواند در آن قرار گیرد را برمی‌گرداند.

تابع MiniMax:

در این تابع، با دریافت نقشه محیط، عمق، آلفا، بتا، و بازیکن الگوریتم MiniMax به همراه هرس آلفا و بتا پیاده‌سازی شده است. ابتدا بررسی می‌کنیم آیا نقشه دریافی یک برگ از درخت بازی با عمق ۵ است یا خیر. در صورتی که به حداکثر عمق مشخص شده رسیده باشیم، امتیاز آن گره را با استفاده از تابع هیوریستیک محاسبه کرده و با کنش None برمی‌گردانیم. در صورتی که به یک برگ رسیده باشیم، با توجه به اینکه عامل هوشمند یا بازیکن انسان برنده شده باشد، امتیاز حداکثر و حداقلی تعیین شده به همراه کنش None برگردانده می‌شود.

در صورتی که به برگ‌ها یا حداکثر عمق تعیین شده نرسیده باشیم، به ادامه الگوریتم MiniMax می‌پردازیم. اگر بازیکن عامل هوشمند باشد، حریف را بازیکن انسان تعیین کرده و ارزش را برابر منفی بینهایت قرار می‌دهیم. ستون‌هایی که قابلیت گذاشتن مهره دارند را پیدا کرده و برای مقداردهی اولیه متغیر بهترین حرکت یکی از ستون‌های معتبر را به صورت تصادفی انتخاب می‌کنیم. یک حلقه روی

ستون‌های معتبر زده و سطری از آن ستون که مهره می‌تواند روی آن قرار بگیرد را پیدا می‌کنیم. پس از آن، یک کپی از نقشه گرفته و روی سطر و ستونی که داریم عدد مربوط به عامل هوشمند را قرار می‌دهیم و برای به دست آوردن امتیاز آن حالت از محیط مجدد تابع MiniMax را فرخوانی کرده و تاجایی پیش می‌رود که به برگ یا حداقل عمق مشخص شده برسد و به صورت بازگشتی امتیاز را محاسبه کرده و برمی‌گرداند. امتیاز به دست آمده از این حالت محیط با بهترین امتیازی که تا حالا داریم مقایسه شده و در صورتی که امتیاز به دست آمده بیشتر باشد، این امتیاز را برابر با بهترین امتیاز قرار داده و بهترین حرکت را برابر ستونی قرار می‌دهیم که این بهترین امتیاز از آن به دست آمده است. حال برای به کارگیری هرس آلفا و بتا، از آن‌جایی که در گره max هستیم، آلفا را محاسبه کرده و شرط هرس درخت را بررسی می‌کنیم در صورتی که این شرط برقرار بود، سایر شاخه‌های درخت را هرس می‌کنیم. در نهایت نیز، بهترین حرکت و امتیاز را برای آن حالت از محیط برمی‌گردانیم.

اگر بازیکن برابر بازیکن انسان باشد، تمامی مراحل بالا را طی کرده، اما در قسمت هرس آلفا و بتا، چون در گره min هستیم، بتا را محاسبه کرده و شرط هرس درخت را بررسی می‌کنیم؛ در صورتی که این شرط برقرار بود، سایر شاخه‌های درخت را هرس می‌کنیم. در نهایت، بهترین حرکت و امتیاز را برای آن حالت از محیط برمی‌گردانیم.

قسمت main برنامه:

در ابتدا فولدر steps_pic را ایجاد کرده تا هر ply از بازی را به صورت عکس در این فولدر ذخیره کنیم. پس از آن، در یک حلقه for، ابتدا وضعیت محیط را گرفته و در صورتی که بازی به پایان نرسیده باشد و نوبت عامل هوشمند باشد، الگوریتم MiniMax را اجرا کرده و حرکت مناسب را به دست می‌آوریم و برای اینکه تمرکز عامل از خانه‌های وسط کاسته شود، متغیر اپسیلون را ضربدر ۰/۹ می‌کنیم. در صورتی که نوبت بازیکن انسان باشد، یک شماره ستون از کاربر گرفته و اگر شماره ستون معتبر باشد آن را به عنوان حرکت بازیکن ذخیره می‌کند. در نهایت حرکت را روی محیط اعمال کرده و صفحه بازی را به صورت عکس ذخیره می‌کنیم.

تحلیل نتایج

با افزایش مقدار عمق، زمانی که طول می‌کشد تا عامل هوشمند یک کنش انتخاب کند افزایش یافته؛ زیرا باید درختی با عمق بیشتر بسازد و امتیاز تمامی گره‌های ساخته شده را محاسبه کند.

منابع ایده

از منابع زیر برای آشنایی با نحوه پیاده‌سازی و الگو گرفتن اولیه استفاده شده است.

Stuart J. Russell, Peter Norvig - Artificial Intelligence_ A Modern Approach, Global Edition-Pearson (2021)

KeithGalli/Connect4-Python: Connect 4 programmed in python using pygame (github.com)