دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده ریاضی و علوم کامپیوتر

پروژه سوم هوش مصنوعی
رشته علوم کامپیوتر

الگوریتم Minimax

نگارش
علیرضا مختاری

استاد درس
مهدی قطعی

استاد کارگاه
بهنام یوسفی مهر

مهر 1403

چکیده

تمرین حاضر به پیاده‌سازی بازی دو نفره Othello با استفاده از الگوریتم‌های هوش مصنوعی می‌پردازد. در این بازی، هر بازیکن باید دیسک‌های حریف را در یک خط محاصره کند تا رنگ آن‌ها را تغییر دهد. الگوریتم‌های Minimax و Expectimax برای تصمیم‌گیری در این بازی استفاده می‌شوند. علاوه بر این، از تکنیک هرس آلفا-بتا برای بهینه‌سازی الگوریتم Minimax استفاده شده است. پروژه شامل کلاس‌ها و توابع مختلفی مانند کلاس‌های مدیریت تخته، بازیکنان و بازی است. همچنین معیارهای ارزیابی مختلفی برای تحلیل عملکرد بازی و انتخاب بهترین حرکت معرفی شده است.

واژه‌های کلیدی:

Othello، Minimax، Expectimax، هرس آلفا-بتا، هوش مصنوعی، تابع ارزیابی، تصمیم‌گیری، بازی دو نفره، الگوریتم بهینه‌سازی.

فهرست مطالب

چکیده ..... أ

فصل اول مقدمه **3**

فصل دوم Minimax و Expectimax در Othello **4**

2-1- توضیح کلی و نحوه پیاده سازی بازی 5

2-3-1- توضیح کلی از بازی:Othello 5

2-3-2- پیاده‌سازی بازی:Othello 5

2-2- توضیح درمورد الگوریتم 6

2-3- تحلیل عمق درخت 7

فصل سوم جمع‌بندی و نتیجه‌گیری و پیشنهادات **10**

منابع و مراجع **12**

فصل اول

مقدمه

.

فصل دوم

Minimax و Expectimax در Othello

2-1- توضیح کلی و نحوه پیاده سازی بازی

2-1-3- توضیح کلی از بازی:Othello

بازیOthello که با نام Reversi هم شناخته می‌شود یک بازی استراتژیک برای دو نفر است. در این بازی، دو بازیکن به ترتیب دیسک‌های خود را روی تخته 8*8 قرار می‌دهند، که یک طرف دیسک‌ها سیاه و طرف دیگر سفید است. هدف هر بازیکن این است که در پایان بازی بیشترین تعداد دیسک‌ها از رنگ خود را داشته باشد. زمانی که بازیکن دیسک خود را روی تخته قرار می‌دهد، اگر یک یا چند دیسک حریف بین دیسک جدید و دیسک‌های موجود او محاصره شود، آن دیسک‌ها به رنگ بازیکن تغییر می‌کنند. بازی زمانی تمام می‌شود که هیچ حرکت قانونی برای هیچ‌کدام از بازیکنان باقی نماند.

2-3-2- پیاده‌سازی بازی:Othello

برای پیاده‌سازی این بازی، از چندین کلاس و متودهای کلیدی استفاده شده است تا مدیریت بازی، قوانین و منطق تصمیم‌گیری را پوشش دهد. در ادامه توضیحی از کلاس‌ها و متودهای مهم این پیاده‌سازی آورده شده است:

2-1-1-1-1- کلاس Board

کلاس `Board`به عنوان هسته اصلی بازی عمل می‌کند. این کلاس وضعیت فعلی تخته، حرکت‌های قانونی و اعمال حرکت‌ها را مدیریت می‌کند. متودهای کلیدی این کلاس عبارتند از:

- **`__init__(self, size)`**: این متود برای ساختن تخته بازی استفاده می‌شود و تخته را با اندازه مشخص) معمولاً ۸x۸ (و دیسک‌های ابتدایی در مرکز صفحه مقداردهی اولیه می‌کند.
- **`get_valid_moves(self, player_color)`** : این متود تمامی حرکت‌های قانونی برای بازیکن مشخص شده را بازمی‌گرداند. در این متود بررسی می‌شود که آیا با قرار دادن دیسک جدید، دیسک‌های حریف محاصره می‌شوند یا خیر.
- **`place_piece(self, row, col, player_color)`** : این متود برای اعمال حرکت استفاده می‌شود و دیسک بازیکن را در موقعیت مشخص شده قرار می‌دهد و دیسک‌های محاصره شده حریف را تغییر رنگ می‌دهد.
- **`get_score(self)`** : این متود تعداد دیسک‌های سیاه و سفید روی تخته را بازمی‌گرداند تا امتیاز هر بازیکن مشخص شود.

1-1-1-2- منطق Minimax و Expectimax:

برای ایجاد یک هوش مصنوعی که بتواند حرکت‌های بهینه را در بازی انتخاب کند، از الگوریتم‌های Minimax و Expectimax استفاده شده است.

- **Minimax :** این الگوریتم به‌صورت بازگشتی تمام حالات ممکن بازی را بررسی می‌کند و بهترین حرکت ممکن را برای بازیکن انتخاب می‌کند. همچنین نسخه بهینه‌تری از این الگوریتم به نام Minimax با هرس آلفا-بتا پیاده‌سازی شده است که سرعت جستجو را افزایش می‌دهد.
- **Expectimax :** این الگوریتم مشابه Minimax است، اما برای بازی‌هایی که دارای احتمالات یا حرکات تصادفی هستند طراحی شده است. در این پیاده‌سازی، از Expectimaxبرای شبیه‌سازی حرکات تصادفی حریف استفاده شده است و بر اساس انتظار، بهترین حرکت انتخاب می‌شود.

1-1-1-3- کلاس Player

کلاس `Player`نماینده هر یک از بازیکنان است. این کلاس می‌تواند یک بازیکن انسانی یا یک بازیکن هوش مصنوعی باشد. متودهای کلیدی این کلاس عبارتند از:

- **`make_move(self, board)` :** این متود برای بازیکنان هوش مصنوعی حرکت را انتخاب می‌کند. با استفاده از الگوریتم‌های Minimax یاExpectimax ، بهترین حرکت انتخاب می‌شود.

1-1-1-4- کلاس Game

کلاس `Game`مدیریت کلی بازی را برعهده دارد. این کلاس وظیفه دارد تا روند کلی بازی از جمله شروع بازی، تبادل نوبت‌ها بین بازیکنان و اعلام پایان بازی را کنترل کند.

- **`play_ai_game(self)` :** این متود بازی را برای بازیکنان هوش مصنوعی اجرا می‌کند و تا پایان بازی روند حرکات را مدیریت می‌کند.

2-2- توضیح درمورد الگوریتم

2-3- تحلیل عمق درخت

عمق درخت تصمیم‌گیری یکی از عوامل کلیدی در عملکرد الگوریتم‌های Minimax و Expectimax است. عمق تعیین می‌کند که الگوریتم تا چه اندازه از حالت‌های آینده بازی را بررسی کند. عمق بیشتر به معنای بررسی تعداد بیشتری از حرکات ممکن است، که می‌تواند به دقت بالاتر منجر شود اما هم‌زمان باعث افزایش زمان و پیچیدگی محاسباتی نیز می‌شود.

انتخاب عمق و علت آن

در انتخاب عمق الگوریتم باید به تعادلی میان دقت و کارایی توجه داشت. انتخاب عمق مناسب به عواملی مانند محدودیت‌های زمانی، پیچیدگی بازی و قدرت پردازشی بستگی دارد. به‌طور کلی، عمق بیشتر اطلاعات بیشتری درباره وضعیت‌های آینده بازی ارائه می‌دهد، اما هزینه آن افزایش نمایی در تعداد حالت‌های بررسی‌شده است.

عمق 1: الگوریتم فقط حرکت بعدی را ارزیابی می‌کند. تصمیمات در این حالت بسیار سطحی و کوتاه‌مدت هستند و معمولاً منجر به بازی‌های ضعیف‌تری می‌شوند، زیرا بازیکن آینده حرکات حریف را در نظر نمی‌گیرد.

عمق 2: الگوریتم حرکت خود و حرکت بعدی حریف را بررسی می‌کند. در این حالت، بازیکن علاوه بر حرکت خود، به عکس‌العمل حریف نیز توجه می‌کند و سعی می‌کند حرکاتی را که باعث بهبود وضعیت حریف می‌شوند، شناسایی کند و از آن‌ها اجتناب کند.

عمق 3: الگوریتم حرکت خود، حرکت حریف و سپس حرکت بعدی خود را بررسی می‌کند. در این عمق، بازیکن نه تنها واکنش حریف را بررسی می‌کند، بلکه به این فکر می‌کند که پس از پاسخ حریف، چه موقعیتی می‌تواند به دست آورد. این باعث می‌شود تصمیمات عمیق‌تر و استراتژیک‌تری گرفته شود.

تجزیه و تحلیل عملکرد در عمق‌های مختلف

الف. عملکرد در عمق 2

در عمق 2، الگوریتم دو حرکت به جلو را بررسی می‌کند: حرکت فعلی بازیکن و پاسخ حریف. این باعث می‌شود بازیکن بتواند حرکاتی را انتخاب کند که نه‌تنها در حال حاضر خوب هستند، بلکه جلوی حرکات قوی حریف را نیز بگیرند. با این حال، چون تنها دو حرکت بررسی می‌شود، بازیکن هنوز از حرکات بلندمدت و استراتژیک قوی برخوردار نیست و ممکن است نتواند به خوبی در برابر حریفی که عمیق‌تر فکر می‌کند مقاومت کند.

مزایا: سریع‌تر اجرا می‌شود و در زمان کوتاه‌تری نتایج معقولی می‌دهد.

معایب: تصمیمات هنوز به‌اندازه کافی عمیق و هوشمندانه نیستند و بازیکن در برابر حرکات پیچیده‌تر آسیب‌پذیر است.

ب. عملکرد در عمق 3

در عمق 3، الگوریتم به بررسی یک حرکت بیشتر می‌پردازد. این به بازیکن امکان می‌دهد که استراتژی‌های پیچیده‌تری ایجاد کند و نه‌تنها واکنش حریف، بلکه واکنش به پاسخ حریف را نیز در نظر بگیرد. این عمق معمولاً تعادلی خوب بین دقت و پیچیدگی زمانی است، زیرا بازیکن به خوبی می‌تواند حرکات را پیش‌بینی کند و هم‌زمان به اندازه کافی از حرکات حریف اطلاع دارد.

مزایا: تصمیمات بسیار استراتژیک‌تر و هوشمندانه‌تر هستند. بازیکن می‌تواند حرکات پیچیده‌تری انجام دهد و در برابر حریفان با عمق کمتر به‌خوبی عمل کند.

معایب: زمان اجرا به طور قابل توجهی افزایش می‌یابد، به خصوص در بازی‌های پیچیده مانند Othello، که در هر نوبت ممکن است حرکات بسیاری وجود داشته باشد.

3. تجزیه و تحلیل تغییرات عملکرد

دقت و قدرت تصمیم‌گیری: هر چه عمق بیشتر شود، تصمیمات بازیکن دقیق‌تر و استراتژیک‌تر می‌شود، زیرا الگوریتم می‌تواند حرکات بعدی را بهتر پیش‌بینی کند. در عمق 1، بازیکن فقط بر اساس حرکات فعلی تصمیم‌گیری می‌کند و عملاً آینده بازی را نمی‌بیند. در عمق 2، بازیکن می‌تواند به صورت محدود حرکات حریف را پیش‌بینی کند. در عمق 3، بازیکن با در نظر گرفتن حرکات چندین نوبت آینده، می‌تواند استراتژی‌های پیچیده‌تری پیاده‌سازی کند.

زمان اجرا: افزایش عمق باعث افزایش نمایی تعداد حالت‌های بررسی‌شده می‌شود. در عمق 1، الگوریتم فقط حرکات فعلی را ارزیابی می‌کند، بنابراین سرعت بسیار بالایی دارد. در عمق 2، الگوریتم باید تمامی پاسخ‌های ممکن حریف را نیز بررسی کند که باعث افزایش تعداد حالت‌ها می‌شود. در عمق 3، هر حرکت بازیکن، هر حرکت حریف و سپس هر حرکت بعدی بازیکن باید بررسی شود، که منجر به افزایش قابل توجه زمان محاسبات می‌شود.

هرس آلفا-بتا: در Minimax با استفاده از هرس آلفا-بتا، می‌توان بسیاری از شاخه‌های درخت تصمیم‌گیری را هرس کرد و از بررسی بی‌فایده آن‌ها اجتناب کرد. این تکنیک باعث می‌شود که حتی در عمق‌های بالاتر، زمان اجرا کاهش یابد، زیرا نیازی به بررسی تمامی حالت‌ها نیست. بنابراین در عمق‌های بالا، هرس آلفا-بتا تأثیر مثبتی بر سرعت الگوریتم دارد.

عمق 2 در Minimax یا Expectimax برای بازی‌هایی که نیاز به زمان پاسخ سریع دارند مناسب است، اما اگر زمان کافی برای محاسبات وجود داشته باشد، عمق 3 تصمیمات بهتری را فراهم می‌کند. با افزایش عمق، بازیکن می‌تواند حرکات پیچیده‌تر و استراتژیک‌تری انجام دهد، اما این کار به هزینه افزایش زمان اجرا و پیچیدگی محاسباتی تمام می‌شود.

فصل سوم

جمع‌بندی و نتیجه‌گیری و پیشنهادات

جمع بندی و نتیجه گیری

این پروژه نشان داد که استفاده از الگوریتم‌های هوش مصنوعی در بازی‌های دو نفره مانند Othello به عامل هوشمند این امکان را می‌دهد که تصمیمات بهتری بگیرد و با تحلیل چند حرکت آینده، بازی را به صورت بهینه مدیریت کند. همچنین استفاده از بهینه‌سازی‌هایی مانند هرس آلفا-بتا می‌تواند به طور قابل توجهی سرعت و کارایی الگوریتم‌ها را بهبود بخشد.

منابع و مراجع

[1] Stuart Russell, Peter Norvig, "Artificial Intelligence: A Modern Approach," 4th Edition, Pearson, 2020.

[2] https://www.geeksforgeeks.org/

[3] https://stackoverflow.com/

[4] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). "A formal basis for the heuristic determination of minimum cost paths".