مدرس:
دکتر فدایی و دکتر یعقوب‌زاده

طراحان:
ژنتیک: سید پارسا حسینی‌نژاد، آرش رسولی، شایان شبیهی
بازی: کیانوش عرشی، امیر فراهانی، طاها فخاریان

مهلت تحویل: 18 آذرماه 1401، ساعت 23:59

قسمت‌اول: ژنتیک

مقدمه

در این پروژه، با روش‌هایی که برگرفته از طبیعت و انتخاب طبیعی هستند، آشنا می‌شویم. در این روش‌ها که به طور کلی الگوریتم‌های ژنتیک نامیده می‌شوند، ایده‌هایی برای مدل‌سازی جفت‌گیری، جهش و انتخاب طبیعی به کار گرفته می‌شود. در این گونه الگوریتم‌ها، ممکن است با انتخاب معیارهای ساده‌ی انتخاب طبیعی، نتایج مطلوب به دست نیاید و باید معیاری در نظر بگیریم که علاوه بر عملکرد فردی، به گوناگونی جمعیت نیز اهمیت دهد.
الگوریتم‌های ژنتیک عموماً در مسئله‌هایی با فضای حالت بزرگ کاربرد دارند؛ این الگوریتم‌ها این کار را با نمونه گرفتن از جمعیت و ترکیب و تغییر افراد و ارزیابی آن‌ها انجام می‌دهند و سعی می‌کنند که نسل به نسل جواب‌ها را بهبود دهند تا به جواب مورد نظر برسند.
در این پروژه با استفاده از الگوریتم‌های ژنتیک، مسئله‌ی معادله‌ی برابری (Equation Problem یا EP) را پیاده سازی کنیم.

توضیح پروژه

هدف از این پروژه آشنایی با الگوریتم‌های ژنتیک در هوش مصنوعی و پیاده سازی یک الگوریتم برای حل EP بر پایه این نوع از الگوریتم‌ها می‌باشد. به بیان دیگر، نیاز است الگوریتمی طراحی کنید که با بکارگیری روش های کلی در الگوریتم‌های ژنتیک، موسوم به crossover و mutation، بر پایه توضیحات ارائه شده در درس به حل EP بپردازید.
در EP، هدف تعیین تعداد مشخصی از عملگر و عملوندهایی است که با قرارگیری این عناصر در جایگاه‌های متناظر در ساختار معادله، برابری طرفین معادله برقرار گردد. به طور مثال، با فرض داشتن مجموعه عملوندهای ورودی شامل اعداد $[3, 4, 6, 7]$، مجموعه عملگرهای قابل استفاده شامل عملگرهای جمع و باقی‌مانده، تعداد 5 جایگاه قابل پر کردن در ساختار معادله، و جواب نهایی 7 در سمت راست معادله نهایی، یک پاسخ قابل ارائه برای این مسئله بصورت زیر می‌باشد:

$$6_1 +_2 4_3 \%_4 3_5 = 7$$

در اینجا، شماره جایگاه‌ها در ساختار معادله با اندیس‌هایی برای عناصر پر کننده نمایش داده شده است.

در این پروژه، انتظار میرود برنامه شما **به ترتیب** تعداد جایگاه‌های ساختار معادله اولیه، لیستی از اعداد مثبت طبیعی یک یا چند رقمی به عنوان مجموعه عملوندهای قابل استفاده، لیستی از عملگر یا عملگرهای قابل استفاده شامل عملگرهای جمع (+)، تفریق (-)، ضرب (*)، مود یا باقیمانده (%)، و در نهایت عدد سمت راست معادله مقصد را در ورودی دریافت کرده، و با استفاده از روش مذکور، در خروجی به ارائه یکی از پاسخ‌های مسئله EP حاصل بپردازد. لازم به ذکر است که می‌بایست بین هر دو عملوند دقیقا یک عملگر از لیست ارائه شده استفاده شود. دقت کنید عدد دسیمال سمت راست معادله حداکثر ۸ رقم دارد. **فرض کنید معادله حتماً جواب دارد.**

در زیر یک نمونه ورودی و خروجی برای مسئله مذکور ارائه میگردد. لازم به ذکر است که برای دریافت ورودی در کد خود می‌توانید از تابع *input* در پایتون استفاده کرده، و یا این ورودی ها را از طریق CLI دریافت کنید. همچنین می‌توانید متغیرها را در بدنه‌ی برنامه مقداردهی کنید.

ورودی:

```
21
1 2 3 4 5 6 7 8
+ - *
18019
```

خروجی:

```
4+7*8*8*8*5-8+7+3*8*4
```

پیاده‌سازی مسئله

بخش یک: مشخص‌کردن مفاهیم اولیه

در الگوریتم‌های ژنتیک ابتدا باید یک تعریف برای ژن ارائه دهید و سپس با استفاده از آن، یک کروموزوم بسازید. هر کروموزوم مجموعه‌ای از ژن‌ها است و این مجموعه یا همان کروموزوم، یک راه پیشنهادی برای حل مسئله مورد نظر می‌باشد. توجه داشته باشید که در الگوریتم‌های ژنتیک باید اکثر کارها را با استفاده از تصادفی کردن وقایع انجام دهید، چرا که اگر فضای حالت بزرگ باشد پیدا کردن شرطی که همه‌ی محدودیت‌ها را برقرار سازد بسیار دشوار است. به همین دلیل، تعریف کروموزوم ها اهمیت ویژه ای دارد و باید به گونه ای باشد که امکان اعمال تابع تناسب و توابع دیگر بر روی آن فراهم باشد.

بخش دو: تولید جمعیت اولیه

پس از تعریف و پیاده‌سازی کروموزوم ها، باید جمعیت اولیه‌ای از کروموزوم‌ها به صورت تصادفی بسازید. تعداد این جمعیت می‌تواند به عنوان یک پارامتر حل مسئله باشد و به انتخاب‌های شما بستگی دارد.

بخش سه: پیاده‌سازی و مشخص‌کردن تابع معیار سازگاری[1]

بعد از تولید جمعیت اولیه، نیاز داریم تا تابع معیاری تعریف کنیم که بتواند برای شناسایی کروموزوم‌های برتر که شرایط و محدودیت‌های مسئله را بهتر مدل می‌کنند استفاده شود. ابتدا یک تعریف مناسب برای این تابع معیار ارائه دهید، و سپس آن را برای این مسئله پیاده سازی کرده، و میزان سازگاری جمعیت خود را بدست آورید.

بخش چهار: پیاده‌سازی crossover و mutation و تولید جمعیت بعدی

حال برای اینکه به یک پاسخ از مسئله داده شده نزدیک شویم، نیاز است در هر نسل، جمعیت جدیدی از جمعیت نسل قبل آن تولید گردد. برای این کار، باید از روش های مذکور موسوم به crossover و mutation استفاده گردد، که در درس با آنها آشنا شده اید.

تابع crossover بر روی دو کروموزوم اعمال می شود، و آن‌ها را ترکیب می‌کند تا به کروموزوم هایی از ترکیب آن دو که در حالت ایده آل بهترین ویژگی های دو ژن اولیه را دارند برسد. این ترکیب و نرخ ایجاد آن باید به عنوان پارامترهای مسئله در نظر گرفته شوند.

تابع mutation بر روی یک کروموزوم اعمال می‌شود، و آن را جهش و یا تغییر می‌دهد؛ به امید آن که بتواند به کروموزوم بهتری دست یابد. می‌توانید درصد معقولی از ژن‌های برتر را نیز برای انتقال مستقیم به نسل‌های آینده در نظر بگیرید.

بخش پنج: ایجاد الگوریتم ژنتیک روی مسئله

در آخر باید این توابع پیاده‌سازی شده را در یک الگوریتم استفاده کنید. توجه کنید که نیاز است هایپر پارامترهایی برای میزان randomness و نحوه نزدیک شدن به پاسخ نهایی خود داشته باشید که با تغییر آن ها به جواب بهتری برسید.

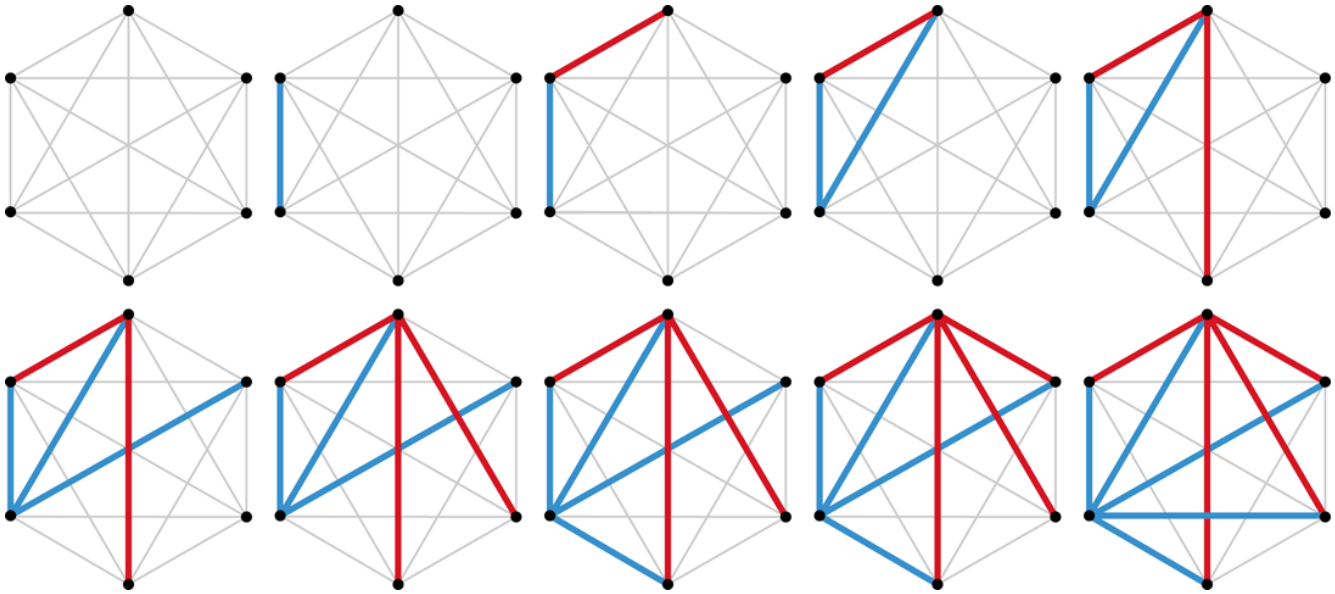
بخش شش: سوالات

1. جمعیت اولیه‌ی بسیار کم یا بسیار زیاد چه مشکلاتی را به وجود می‌آورند؟
2. اگر تعداد جمعیت در هر دوره افزایش یابد، چه تاثیری روی دقت و سرعت الگوریتم می‌گذارد؟
3. تاثیر هر یک از عملیات‌های crossover و mutation را بیان و مقایسه کنید. آیا می‌توان فقط یکی از آنها را استفاده کرد ؟ چرا ؟
4. به نظر شما چه راهکارهایی برای سریعتر به جواب رسیدن در این مسئله‌ی خاص وجود دارد؟
5. با وجود استفاده از این روش‌ها، باز هم ممکن است که کروموزوم‌ها پس از چند مرحله دیگر تغییر نکنند. دلیل این اتفاق و مشکلاتی که به وجود می آورد را شرح دهید. برای حل آن چه پیشنهادی می‌دهید؟ (راه حلهای خود را امتحان کنید و بهترین آن‌ها را روی پروژه خود پیاده سازی کنید.)
6. چه راه حلی برای تمام شدن برنامه در صورتی که مسئله جواب نداشته باشد پیشنهاد می‌دهید؟

[1] Fitness Function

قسمت‌دوم: بازی

مقدمه



پس از فراگیری مباحث هوش مصنوعی و آشنا شدن با بازی Sim، شما و دوستتان تصمیم گرفته‌اید بر روی agent ای برای بازی کردن Sim کار کنید و پس از مدتی، آن دو agent را مجبور به رقابت باهم کنید و برنده نهایی را از آنجا شناسایی کنید. شما که به هیچ‌وجه نمی‌خواستید به دوستتان ببازید، تصمیم گرفتید که با استفاده از الگوریتم min-max یک agent ای بنویسید که یک بار برای همیشه برنده را بین خود و دوستتان مشخص کنید.

توضیح بازی

هدف در بازی Sim، مجبور کردن رقیب به ساختن یک مثلث هم‌رنگ در بین خط‌های واصل بین نقاط موجود در صفحه است. به عبارتی، اولین بازیکنی که مجبور شود که یک مثلث را به رنگ خود در صفحه بازی کامل کند، بازنده بازی است.

صفحه بازی Sim شامل ۶ نقطه است و در هر نوبت، بازیکن یک خط بین دو نقطه از این ۶ نقطه را به رنگ خود در می‌آورد. باید توجه کرد که در نسخه ۶ نقطه‌ای این بازی، امکان تساوی وجود ندارد، زیرا هیچ راهی نیست که تمامی خط‌ها را رنگ کرد، بدون اینکه حداقل یک مثلث ساخته شود.

پیاده سازی

هدف شما پیاده سازی الگوریتم min-max برای شکست دادن دوستتان است. کد بازی در فایل main.py به شما داده شده است اما این کد کامل نیست و شما باید بخش‌های TODO را کامل کنید. شما باید تابع minimax را کامل کنید که در واقع پیاده‌سازی الگوریتم min-max برای این بازی است.

شما می‌توانید برای تمیزی کد خود، متد و توابع دیگری را به کد اضافه کنید اما حق هیچ‌گونه تغییری در بخش‌های دیگر کد را ندارید و این بخش‌ها باید ثابت باقی بمانند (اضافه کردن مواردی مثل getter و یا setter مانعی ندارد اما باید در گزارش کار ذکر کنید).

دقت کنید که باید برای الگوریتم min-max خود یک تابع heuristic برای ارزشیابی هر یک از حالات تعریف کنید. تابع heuristic خود را در گزارش شرح دهید.

همچنین برای استفاده از رابط گرافیکی به منظور پیاده‌سازی راحتتر، می‌توانید پرچم GUI را True کنید. همچنین برای اجرای برنامه نیز از دستور زیر استفاده کنید:

python3 main.py <minimax_depth> <gui>

بررسی نتایج

برای درک کامل و آزمایش کد خود، با 3 بار ران کردن کد خود برای عمق های 1، 3 و 5، بررسی کنید که شانس پیروزی شما چه مقدار است. همینطور زمان اجرای هر عمق را ثبت کنید.

هرس آلفا و بتا: برای افزایش سرعت کد و کاهش نودهای خود، هرس آلفا و بتا را به کد اضافه کنید و سرعت اجرای کد، تعداد نودهای مورد بررسی و شانس پیروزی را برای عمق‌های یاد شده مجدداً بررسی کنید. همچنین عمق 7 را به عمق‌های مورد بررسی خود اضافه کنید.

نکات تکمیلی:

- نتایج تمامی بخش‌ها را به صورت کامل در گزارش خود بیاورید.
- روش محاسبه شانس پیروزی: بازی را 100 الی 200 مرتبه در حالت مدنظر اجرا کنید و با بدست آوردن تعداد بردها، شانس پیروزی را حساب کنید.

سوالات

- **سوال ۱**: یک heuristic خوب چه ویژگی‌هایی دارد؟ علت انتخاب heuristic شما و دلیل برتری آن نسبت به تعدادی از روش‌های دیگر را بیان کنید.
- **سوال ۲**: آیا میان عمق الگوریتم و پارامترهای حساب شده روابطی می‌بینید؟ به طور کامل بررسی کنید که عمق الگوریتم چه تاثیراتی بر روی شانس پیروزی، زمان و گره‌های دیده شده می‌گذارد.

- **سوال ۳**: وقتی از روش هرس کردن استفاده می‌کنید، برای هر گره درخت، فرزندانش به چه ترتیبی اضافه می‌شوند؟ آیا این ترتیب اهمیت دارد؟ چرا این ترتیب را انتخاب کردید؟

نکات پایانی

- نتایج و گزارش‌های خود را در یک فایل فشرده با عنوان AI-CA2-Genetics-Game-SID.zip آپلود کنید.
- یک **شبه کد** برای قسمت ژنتیک به نام CA2.py به همراه صورت پروژه بارگذاری شده است. شما می‌توانید از این شبه کد کمک بگیرید و قسمت‌هایی از آن را که با عبارت #TODO علامت گذاری شده‌اند پر کنید و کد را تکمیل کنید. توجه کنید استفاده از این شبه کد **اختیاری** است و می‌توانید پیاده‌سازی خود را داشته باشید.
- توجه داشته باشید علاوه بر ارسال فایل‌های پروژه، این پروژه تحویل نیز گرفته خواهد شد. بنابراین لازم است بر تمامی قسمتهای کدتان تسلط کافی داشته باشید و تمام بخش‌های پروژه باید قابلیت اجرای مجدد در زمان تحویل را داشته باشند. همچنین در صورت عدم حضور در زمان تحویل، نمره‌ای دریافت نخواهید کرد.
- هیچگونه شباهتی در انجام این پروژه بین افراد مختلف پذیرفته نمی‌شود. در صورت کشف هرگونه تقلب برای همه افراد متقلب نمره ۱۰۰- در نظر گرفته می‌شود.
- استفاده از مراجع با ارجاع به آنها بلامانع است. اما در صورتی که گزارش شما ترجمه عینی از آنها باشد یا از گزارش افراد دیگر استفاده کرده باشید کار شما تقلب محسوب می‌شود.
- در صورت هرگونه سوال بهتر است در فروم درس مطرح کنید تا بقیه هم از آن استفاده کنند، در غیر این‌صورت با طراحان در ارتباط باشید.
- هدف از تمرین یادگیری شماست. لطفاً خودتان انجام دهید.