هوش مصنوعی

تمرین شماره ۲

مدرسین: **دکتر فدایی و دکتر یعقوب‌زاده**

طراحان: کوروش سجادی، پارسا دقیق، یاسمن عموجعفری

مهلت تحویل: شنبه ۲۱ آبان ۱۴۰۴، ساعت ۲۳:۵۹

قسمت اول : الگوریتم ژنتیک

اسکوبی‌دو و دوستانش، به جای پیدا کردن ارواح، این بار به یک چالش جدید برخورده‌اند: یک پازل جورچین عظیم که قطعاتش در یک آشفتگی کامل قرار دارند! . اسکوبی‌دو می‌خواد با قدرت "هوش مصنوعی" (که البته خودش زیاد ازش سر در نمیاره!) این معما رو حل کنه و قدرت الگوریتم ژنتیک رو به همه نشون بده.

توضیح و عمیق‌تر شدن در مسئله

حل پازل جورچین، معمایی در حد یک پرونده‌ی فوق سری برای اسکوبیدو است! ما با یک چالش بزرگ روبرو هستیم: درهم‌ریختگی کامل قطعات پازل، هزاران بار با جادوگریِ تصادفی جابجا شده‌اند و هدف ما، یافتن یک **جایگشت (Permutation)** است که تصویر نهایی را بازسازی کند. این مسئله بهینه‌سازی، به‌طور طبیعی برای الگوریتم ژنتیک مناسب است، زیرا ما به دنبال یک راه‌حل (کروموزوم) در فضای حالت بزرگی از جایگشت‌های ممکن هستیم که معیار **تناسب (Fitness)** آن (یعنی میزان چسبیدن قطعات به یکدیگر) به حداکثر برسد.

صورت مسئله

هدف: پیدا کردن جایگشت بهینه از $N = R \times C$ قطعه‌ی پازل که در یک شبکه‌ی R ردیف و C ستون قرار می‌گیرند، به‌طوری که کمترین میزان تفاوت (Dissimilarity) در مرزهای قطعات مجاور ایجاد شود.

داده‌های ورودی:

- یک تصویر اصلی که به N قطعه‌ی مربعی هم‌اندازه تقسیم شده است.

- **ماتریس تفاوت (Dissimilarity Matrix):** یک ماتریس که میزان تفاوت هر جفت قطعه‌ی پازل را برای دو حالت مجاورت افقی (**چپ-راست**) و عمودی (**بالا-پایین**) ذخیره می‌کند. این معیار معمولاً با استفاده از مجموع مربعات اختلاف پیکسل‌های لبه‌های مجاور (SSD) محاسبه می‌شود.

محدودیتها

ساختار قطعات: فرض می‌شود تمام قطعات پازل دارای اندازه یکسان و شکل مربعی هستند.
جهت‌گیری (Orientation): برای سادگی مسئله، قطعات پازل در طول اجرای الگوریتم قابلیت **چرخش یا برعکس شدن ندارند** و جهت‌گیری آن‌ها ثابت فرض می‌شود (همانطور که در محاسبات `ImageAnalysis` تنها دو جهت "L-R" و "T-D" بررسی شده است).
معیار همگرایی: الگوریتم باید در زمان معقول به یک جواب با **تناسب بالا (Fitness)** برسد.

پیاده سازی مسئله

بخش یک : تابع عدم شباهت (Dissimilarity Function)

قبل از هر چیز، برای سنجش کیفیت هر آرایش پازل، باید راهی برای اندازه‌گیری میزان **"خرابی" یا تفاوت (Dissimilarity)** در مرزهای اتصال قطعات مجاور داشته باشیم. این معیار به ما می‌گوید که چقدر یک قطعه با همسایه‌ی خود همخوانی ندارد. شما موظف هستید که منطق محاسبه **مجموع مربعات اختلاف (Sum of Squared Differences - SSD)** را برای پیکسل‌های لبه‌های مجاور دو قطعه (برای حالات مجاورت افقی و عمودی) پیاده‌سازی کنید. هرچه این مقدار SSD کمتر باشد، نشان‌دهنده همخوانی بهتر لبه‌ها و جفت‌شدن مطلوب‌تر قطعات است. این تابع، هسته‌ی اصلی ارزیابی کیفیت در کل الگوریتم شماست و در کلاس `ImageAnalysis` تکمیل می‌شود.

بخش دو: فرد و تابع سازگاری (Individual & Fitness)

فرد (Individual): یک شی جامع است که حاوی داده‌هایی مانند تصاویر قطعات پازل و ابعاد شبکه ($R \times C$) می‌باشد.

کروموزوم: بخش اصلی فرد، کروموزوم است که نقشه چیدمان قطعات را کد می‌کند. کروموزوم یک آرایه‌ی یک‌بعدی حاوی جایگشت کامل شناسه‌های یکتای تمام قطعات پازل می‌باشد. ترتیب این شناسه‌ها در آرایه، موقعیت دقیق سطر و ستون هر قطعه را در پازل نهایی مشخص می‌کند. شما باید منطق ساختاردهی این کروموزوم و نگهداری ابعاد $R \times C$ را تکمیل کنید.

تابع سازگاری (Fitness Function): برای اینکه بفهمیم کروموزوم چقدر خوب عمل کرده، باید تابع سازگاری آن را محاسبه کنیم. چون هدف ما کمینه کردن عدم شباهت کل است و الگوریتم ژنتیک به دنبال بیشینه‌سازی، تابع فیتنس باید معکوس مجموع کل عدم شباهت‌ها در تمام مرزهای پازل باشد:

$$Fitness = \frac{1}{\sum_{All\ Borders} Dissimilarity}$$

بخش سه: استراتژی انتخاب (Selection Strategy)

برای تولید نسل بعد، ما والدینی را از نسل فعلی انتخاب می‌کنیم که فیتنس بالاتری دارند. شما باید روش انتخاب چرخ رولت (Roulette Wheel Selection) را پیاده‌سازی کنید که در آن شانس انتخاب هر فرد

متناسب با سهم فیتنس او از کل جمعیت است. همچنین، لازم است یک روش selection دیگر را نیز پیاده‌سازی کرده و در گزارش خود تأثیر هر روش انتخاب را بر همگرایی الگوریتم تحلیل کنید.

بخش چهارم: پیاده سازی چرخه‌ی الگوریتم ژنتیک

شما موظف هستید که متد اصلی الگوریتم را برای مدیریت فرآیند تکاملی در طول نسل‌های متوالی پیاده‌سازی کنید. این فرآیند با تولید اولیه جمعیت به صورت تصادفی آغاز می‌شود. در هر نسل، ابتدا باید افراد را بر اساس امتیاز فیتنس ارزیابی و مرتب‌سازی کنید. سپس، نخبگان (Elites) (بهترین افراد) مستقیماً به نسل بعد منتقل می‌شوند تا بهترین راه‌حل‌های کشف شده حفظ گردند. پس از آن، باقی اعضای نسل جدید باید با انتخاب والدین از جمعیت و اعمال عملگرهای ترکیب (Crossover) و جهش (Mutation) روی کروموزوم‌ها تولید شوند. این چرخه تا زمان برآورده شدن شرایط توقف ادامه می‌یابد و در نهایت، بهترین فرد کل دوران تکامل به عنوان جواب نهایی بازگردانده می‌شود.

بخش پنجم: تحلیل نتایج

برای سنجش درستی الگوریتم چندین عکس در اختیار شما قرار داده شده، الگوریتم را برای هر عکس به ازای تعداد جنریشن های مختلف (10،15،20) و piece_size های مختلف (32،64،128) امتحان کنید و خروجی هر عکس را به ازای هر جنریشن چاپ کنید و در صورت نیاز خروجی نهایی رو به صورت تصویر ذخیره کنید.

سوالات

1. برای یک پازل 8×6، هر کروموزوم نشان می‌دهد که 48 قطعه در چه جاهایی قرار گیرند. با فرض برچسب‌گذاری منحصربه‌فرد برای هر قطعه، فضای کل حالت‌های ممکن را بدست آورید و سپس نتیجه را با نزدیک ترین توان 10 تقریب بزنید.

2. یکی از عکس‌هایی که به عنوان تست برای شما قرار داده شد،عکس صفحه‌ی شطرنج است.آیا الگوریتم ژنتیک شما برای این پازل خوب عمل کرده است؟ اگر خیر علت آن را توضیح دهید.

3. روش‌های Crossover در الگوریتم ژنتیک باید متناسب با ساختار کروموزوم طراحی شوند. در این پروژه، Crossover به صورت یک روش ترکیبی ابتکاری و مبتنی بر همسایگی (Heuristic Crossover) پیاده‌سازی شده است که اتصالات محلی را در اولویت قرار می‌دهد.چرا روش‌های ساده ترکیب (مانند one-point یا two-point Crossover) برای حل مسئله‌ی پازل (که یک مسئله‌ی جایگشتی است) نامناسب یا ناکارآمد هستند؟

4. فرآیند انتخاب نسل بعد در الگوریتم ژنتیک، تعادل ظریفی بین فشار انتخاب (Selection Pressure) و تنوع ژنتیکی (Diversity) ایجاد می‌کند. در مورد الگوریتم ژنتیک پازل، توضیح دهید که چرا فشار انتخاب بالا (یعنی تمرکز شدید بر بهترین‌ها) می‌تواند به همگرایی زودرس منجر شود. سپس بگویید که چگونه استفاده از روش‌هایی مانند انتخاب تورنمنت (Tournament Selection)، با اجازه دادن شانس بقا به افراد ضعیف‌تر، به حفظ تنوع و در نهایت کشف بهتر کمک می‌کند.

5. در یک مسئله‌ی جایگشتی مانند پازل، عملگر جهش (Mutation) معمولاً به صورت جابجایی تصادفی دو قطعه (Swap Mutation) پیاده‌سازی می‌شود. نقش اصلی جهش در این نوع فضای حالت چیست؟ توضیح دهید که اگر نرخ جهش (Mutation Rate) را بسیار پایین در نظر بگیرید، چه خطری الگوریتم شما را تهدید می‌کند (از منظر کیفیت نتایج)، و اگر این نرخ را بسیار بالا در نظر بگیرید، الگوریتم به چه نوع جستجوی ناکارآمدی تبدیل می‌شود.

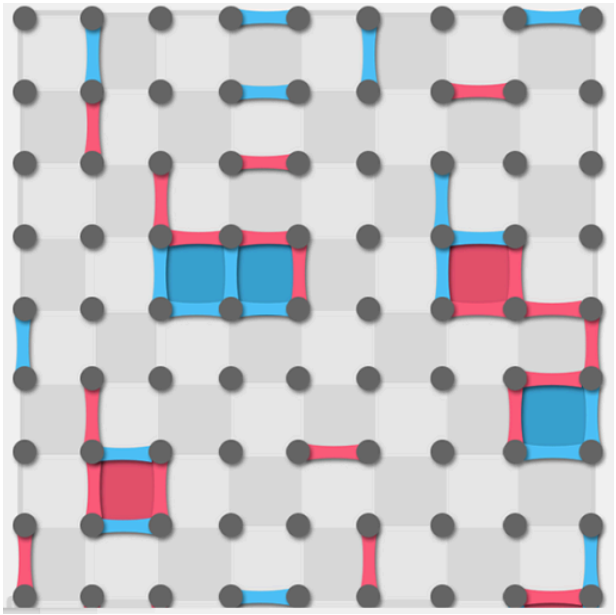
قسمت دوم : Dots and Boxes

اسکوبی‌دو و شگی پس از حل معمای پازل جورچین، خسته و گرسنه به کلبه‌ی مرموز بازگشتند. شگی که حالا با قدرت الگوریتم ژنتیک آشنا شده بود، تصمیم گرفت یک ساندویچ غول‌پیکر درست کند، اما ناگهان یک روح شیطانی ظاهر شد و ساندویچ را به یک شبکه‌ی Dots and Boxes تبدیل کرد! اسکوبی‌دو ترسید و فرار کرد، اما شگی با هوش مصنوعی خودش ایستاد و گفت: «این بار با الگوریتم Minimax این روح رو شکست می‌دم!» از آنجایی که شگی هنوز مهارت برنامه‌نویسی کاملی نداره، دوباره از شما کمک خواسته تا این بازی فکری رو پیاده‌سازی کنه.

توضیح بازی

بازی Dots and Boxes یک بازی دو نفره استراتژیک است که روی یک صفحه شبکه‌ای (Grid) از نقاط انجام می‌شود. بازیکنان به نوبت خطوط افقی یا عمودی بین نقاط مجاور می‌کشند.

- **هدف:** وقتی یک بازیکن چهارمین خط یک مربع را بکشد، آن مربع را مالک می‌شود و امتیاز می‌گیرد. این امتیاز به این معناست که نوبت اضافی دریافت می‌کند.
- **قوانین:** بازیکنان فقط خطوط بین نقاط مجاور را می‌توانند بکشند. بازی وقتی تمام می‌شود که تمام خطوط ممکن کشیده شوند و بازیکن با بیشترین مربع‌های مالک‌شده برنده است.
- **پایان بازی:** اگر تمام مربع‌ها پر شوند، بازیکن با امتیاز بیشتر برنده است. اگر صفحه پر شود و امتیازها برابر باشند، بازی مساوی است.



پیاده سازی

هدف شما پیاده‌سازی الگوریتم Minimax با هرس آلفا-بتا است. کد بازی 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 minimax برای این بازی است. شما می‌توانید برای تمیزی کد خود، متد و توابع دیگری را به کد اضافه کنید اما بهتر است تغییری در بخش‌های دیگر کد ایجاد نکنید و این بخش‌ها ثابت بمانند. برای الگوریتم minimax خود، به یک تابع heuristic برای ارزشیابی هر یک از حالات نیاز دارید. برای استفاده از کد کافیست یک نمونه از کلاس Dots_and_Boxes با توجه به آرگومان‌های مد نظر بسازید و پس از پیاده‌سازی تابع minimax، تابع mainloop را در نمونه‌ی خود صدا زده و بازی شروع می‌شود. می‌توانید از رابط گرافیکی (GUI) برای تست و پیاده‌سازی راحت‌تر استفاده کنید. همچنین برای بررسی صفحه بازی در هر مرحله، می‌توانید از متدهای مربوطه مختص test استفاده کنید.

بررسی نتایج

برای درک کامل الگوریتم، کد را با عمق‌های مختلف (مانند 2، 4، 6) و بدون هرس آلفا-بتا 50 الی 100 بار اجرا کنید و میانگین زمان، شانس پیروزی و میانگین تعداد نودهای دیده شده را برای هر عمق حساب کنید. هرس آلفا و بتا: برای افزایش سرعت کد و کاهش نودهای دیده شده، هرس آلفا و بتا را به کد اضافه کنید و موارد ذکر شده در بخش قبل را مجدداً بررسی کنید. در نهایت نتیجه بازی را برای 100 بار بازی کردن در مقابل حریف رندوم نشان دهید.

در نهایت نیز پاسخ شما و heuristic انتخابی شما توسط TA تحویل گیرنده با استفاده از (GUI) ارزیابی می شود پس مطمئن شوید که به نتیجه مطلوب خود رسیده اید!

سوالات

1. با اجرای آزمایش‌های متعدد، رابطه بین عمق جستجوی Minimax (مانند 2، 4، 6) و معیارهای عملکرد (زمان اجرا، تعداد نودهای ارزیابی‌شده، و نرخ پیروزی) را بررسی کنید. توضیح دهید که افزایش عمق چگونه بر تعادل بین دقت تصمیم‌گیری و کارایی محاسباتی تأثیر می‌گذارد و نقطه بهینه را پیشنهاد دهید.
2. در الگوریتم Minimax با هرس آلفا-بتا، ترتیب بررسی حرکات فرزندان هر نود چقدر بر کارایی هرس تاثیرگذار است؟ روش پیشنهادی در کد (_order_moves) را تحلیل کنید (که حرکات تکمیل‌کننده جعبه، ایمن، و پرریسک را اولویت‌بندی می‌کند). یک استراتژی بهبود یافته پیشنهاد دهید.

3. Branching Factor را توضیح دهید و بگویید که با پیشرفت این بازی چه تغییراتی می‌کند؟

4. با جزئیات توضیح دهید که هرس آلفا-بتا چگونه شاخه‌های غیرضروری درخت بازی را حذف می‌کند بدون اینکه بهترین حرکت را از دست بدهد. با یک مثال ساده از Dots and Boxes (با جعبه 2 در 2) نشان دهید که آلفا و بتا چگونه به‌روزرسانی می‌شوند و تعداد نودهای ارزیابی‌شده را کاهش می‌دهند.

5. در مقابل حریفی که حرکات را به‌صورت کاملاً تصادفی انتخاب می‌کند، Minimax بیش از حد محاسباتی است. چرا؟ الگوریتم جایگزین بهینه‌تری مانند (Monte Carlo Tree Search-MCTS) را پیشنهاد دهید و توضیح دهید چگونه با نمونه‌برداری تصادفی و (Upper Confidence Bound) در چنین سناریوهایی کارآمدتر عمل می‌کند.

نکات پایانی

- دقت کنید که کد شما باید به نحوی زده شده باشد که نتایج قابلیت بازتولید داشته باشند.
- توضیحات مربوط به هر بخش از پروژه را بطور خلاصه و در عین حال مفید در گزارش خود ذکر کنید. از ابزارهای تحلیل داده مانند نمودارها استفاده کنید. حجم توضیحات گزارش شما هیچ گونه تاثیری در نمره نخواهد داشت و تحلیل و نمودارهای شما بیشترین ارزش را دارد.
- سعی کنید از پاسخ‌های روشن در گزارش خود استفاده کنید و اگر پیش‌فرضی در حل سوال در ذهن خود دارید، حتما در گزارش خود آن را ذکر نمایید.
- پس از مطالعه کامل و دقیق صورت پروژه، در صورت وجود هرگونه ابهام یا سوال با طراحان پروژه در ارتباط باشید.
- نتایج، گزارش و کدهای خود را در قالب یک فایل فشرده با فرمت AI_CA2_[stdNumber].zip در سامانه ایلرن بارگذاری کنید. به طور مثال AI_CA2_810100999.zip
- محتویات پوشه باید شامل فایل پاسخ‌های شما به سوالات کتبی، فایل jupyter-notebook، خروجی html و فایل‌های مورد نیاز برای اجرای آن باشد. از نمایش درست خروجی‌های مورد نیاز در فایل html مطمئن شوید.
- توجه کنید این تمرین باید به صورت تک‌نفره انجام شود و پاسخ‌های ارائه شده باید نتیجه فعالیت فرد نویسنده باشد. در صورت مشاهده تقلب به همه افراد مشارکت‌کننده، نمره تمرین 100- و به استاد نیز گزارش می‌گردد. همچنین نوشته نشدن کدها توسط هوش مصنوعی نیز بررسی می‌شود!

موفق باشید