هوش مصنوعی

تمرین شماره ۲

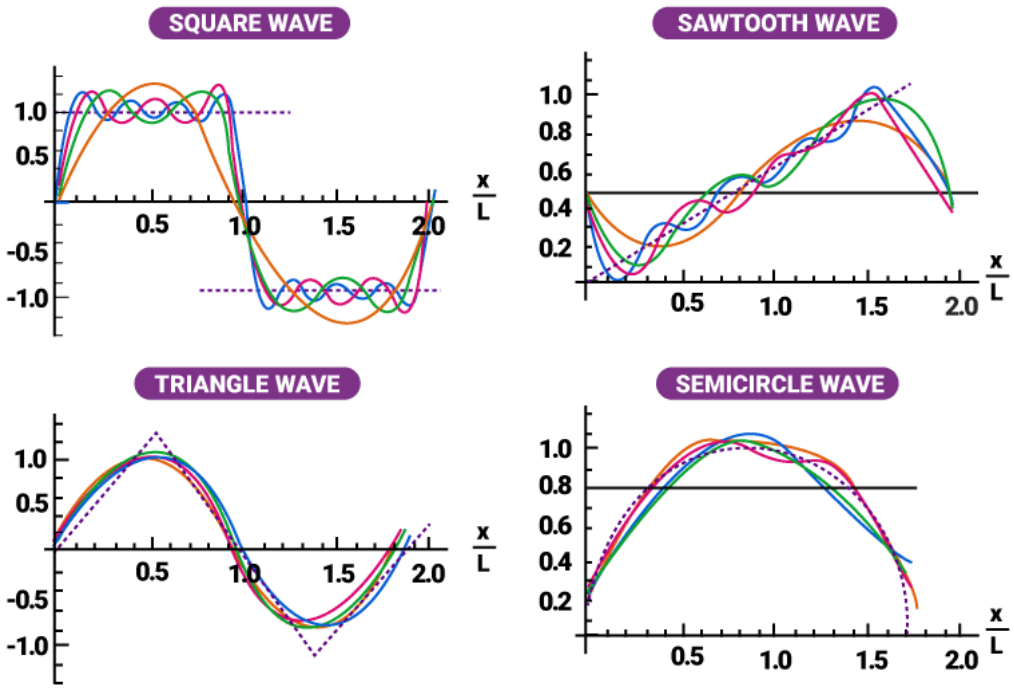
مدرسین: **دکتر فدایی و دکتر یعقوب‌زاده**

طراحان: کوروش سجادی، شهریار عطار، فربد عظیم محسنی

مهلت تحویل: شنبه ۲۳ فروردین ۱۴۰۴، ساعت ۲۳:۵۹

قسمت اول : الگوریتم ژنتیک

متین فردا امتحان ریاضیات مهندسی دارد، که شامل مبحث سری فوریه است. او درک درستی از نحوه محاسبه ضرایب سری فوریه ندارد و نگران است نتواند توابع پیچیده را به اجزای سینوسی و کسینوسی تجزیه کند. او می‌خواهد تا با استفاده از **الگوریتم ژنتیک**، ضرایب سری فوریه یک تابع ناشناخته را به صورت تقریبی پیدا کنید. هدف این است که با شبیه‌سازی فرآیند تکامل، ضرایب بهینه برای بازسازی تابع اصلی در محدوده ۲۰ جمله اول سری فوریه شناسایی شوند. از آنجایی که علاوه بر ریاضیات، متین در برنامه‌نویسی نیز ضعیف است از شما کمک خواسته، به او در انجام پروژه کمک کنید.



توضیح سری فوریه

سری فوریه روشی برای تجزیه یک تابع تناوبی با دوره تناوب (T) به مجموع بی‌پایانی از توابع سینوسی و کسینوسی با فرکانسهای صحیح است. این سری به شکل مقابل تعریف میشود (به شرط اینکه تابع در بازه π − تا π تعریف شده باشد و متناوب باشد):

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos(nx) + b_n \sin(nx) \right]$$

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\, dx$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\, dx, \quad n \geq 1$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\, dx, \quad n \geq 1$$

- a_0: مقدار متوسط تابع در یک دوره تناوب.
- a_n و b_n : ضرایب سینوسی و کسینوسی که با انتگرال‌گیری از تابع اصلی محاسبه میشوند.
- **هدف:** با داشتن نمونه‌هایی از تابع $f(t)$، ضرایب a_n و b_n را طوری بیابیم که سری فوریه تقریب مناسبی از تابع اصلی باشد.

صورت مسئله

متین یک تابع ناشناخته $f(t)$ را در بازه 0 تا T نمونه‌برداری کرده و ۱۰۰ نقطه داده $(t_i , f(t_i))$ جمع‌آوری کرده است. او میخواهد با استفاده از الگوریتم ژنتیک، ضرایب $(a_0 , a_1 , a_2 , \dots , a_{21})$ و $(b_1 , b_2 , \dots , b_{21})$ (مجموعا ۴۱ ضریب) را طوری پیدا کند که بهترین تطابق را با داده‌هایش داشته باشد.

محدودیتها

۱. تنها از **۲۰ جمله اول** سری فوریه استفاده شود (منظور از ۲۰ جمله اول در بخش بالا توضیح داده شده).

۲. ضرایب در بازه $[-A, A]$ محدود شوند، که در ابتدا مقدار آن باید مشخص شود، مثلا A = 10

۳. الگوریتم باید در زمان معقول (که برای هر تست متفاوت است) به جواب برسد. برای فهمیدن رسیدن به جواب، از معیار خطا مانند جذر میانگین مربعات خطا ($RMSE$) و یا بسیاری توابع دیگر که در این زمینه وجود دارد می‌توانید استفاده کنید تا اختلاف تقریب سری فوریه با تابع اصلی را بسنجید.

پیاده سازی مسئله

بخش یک : تعریف مفاهیم

در ابتدای مسئله لازم است تعریف صحیح ژن در مسئله را پیدا کنید و با استفاده از آن‌ها کروموزوم را بسازید. هر کروموزوم یک پاسخ برای مسئله است و با انتخاب کردن کروموزوم‌ها به صورت تصادفی سعی می‌شود که فضای حالت بزرگ مسئله پوشش داده شود.

بخش دو: ساخت جمعیت اولیه

در ابتدا لازم است جمعیتی از کروموزوم‌ها را بسازید. جمعیت شما باعث ایجاد تنوع در حالات شما می‌شود و به شما کمک می‌کند راه حل‌های مختلفی برای رسیدن به پاسخ پیدا کنید و از این رو اهمیت زیادی دارد. دقت کنید که انتخاب پارامتر جمعیت به خود شما بستگی دارد و در گزارش خود باید نمودار مقایسه روند افزایش فیتنس را در سه حالت با جمعیت‌های متفاوت بررسی کنید و نتایج را تحلیل کنید.

بخش سه: معیار سنجش سازگاری[1]

برای اینکه بفهمید الگوریتم شما چه قدر خوب عمل می‌کند باید معیاری برای سنجش سازگاری پیدا کنید. در مسئله داده شده، معیار خوب بودن هر کروموزوم این است که به چه اندازه به تابع هدف نزدیک است. برای سنجش تفاوت دو تابع معیارهای متفاوتی وجود دارد. چند تا از آن‌ها را در گزارش خود بیاورید و در نهایت از یکی از آنها به عنوان معیار سازگاری در الگوریتم خود استفاده کنید همچنین دلیل انتخاب خود را ذکر کنید. توجه داشته باشید که معیار $RMSE$ حتما باید در گزارش شما باشد و به جز آن حداقل باید از دو معیار دیگر نیز استفاده کنید.

بخش چهارم: پیاده سازی crossover، mutation و استراتژی انتخاب نسل بعد

حال برای اینکه به یک پاسخ از مسئله داده شده نزدیک شویم، نیاز است در هر نسل، جمعیت جدیدی با استفاده از جمعیت نسل قبل آن تولید گردد. برای این کار، باید از روش‌های crossover و mutation استفاده گردد. تابع crossover بر روی دو کروموزوم اعمال می‌شود، و آن‌ها را ترکیب می‌کند تا به کروموزوم‌هایی از ترکیب آن دو که در حالت ایده‌آل بهترین ویژگی‌های 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

---

[1] Fitness Evaluation

جهش و یا تغییر می‌دهد؛ به این امید که بتواند به کروموزوم بهتری جهش پیدا کند. می‌توانید درصد معقولی از ژن‌های برتر را نیز برای انتقال مستقیم به نسل‌های آینده در نظر بگیرید.

بخش پنجم: تحلیل نتایج

برای سنجش درستی الگوریتم باید خروجی الگوریتم خودتان را با تابع هدف در یک نمودار رسم کنید و این کار را برای تمام تابع‌های هدفی که در اختیارتان گذاشته شده تکرار کنید. همچنین رسم نمودار فیتنس در نسل‌های متفاوت و تحلیل آن می‌تواند کمک بسیاری به شما بکند.

سوالات

1. برای کروموزومی که در نظر گرفته اید فضای حالت آن را محاسبه کنید.
2. دو تا از ایده‌هایی که از نظر شما می‌تواند باعث سریع‌تر همگرا شدن این مسئله شود را توضیح دهید.
3. استراتژی‌های متفاوتی برای انتخاب نسل بعد در الگوریتم‌های ژنتیک وجود دارد. درباره دو مورد از آن‌ها توضیح دهید.
4. یکی از چالش‌های اصلی در الگوریتم‌های ژنتیک، جلوگیری از همگرایی زودرس (Premature Convergence) است. این پدیده زمانی رخ می‌دهد که جمعیت به سرعت به یک نقطه بهینه محلی همگرا می‌شود و از کشف راه‌حل‌های بهتر باز می‌ماند. دو روش برای جلوگیری از همگرایی زودرس در الگوریتم ژنتیک نام ببرید و توضیح دهید که هر کدام چگونه باعث افزایش تنوع ژنتیکی در جمعیت می‌شوند.
5. یکی از توابع خطایی که در کارهای آماری و مسائل یادگیری ماشین استفاده می‌شود تابع R-squared است. نحوه کار این تابع را توضیح دهید و توضیح دهید آیا در انجام این مسئله می‌تواند کاربردی باشد یا نه.

قسمت دوم : بازی Pentago

متین پس از اینکه موفق شد با هر سختی‌ای بود امتحان ریاضیات مهندسی را بگذراند، به خانه برگشت و تصمیم گرفت به بازی کردن بپردازد. او که به بازی‌های فکری و بازی‌سازی علاقه زیادی داشت تصمیم گرفت تا بازی Pentago را به کمک الگوریتم Minimax پیاده‌سازی کند. از آنجایی که او هنوز مهارت برنامه‌نویسی ندارد، دوباره از شما برای پیاده سازی این بازی کمک خواسته است.

توضیح بازی

بازی Pentago یک بازی دو نفره استراتژیک است که روی یک صفحه ۶×۶ انجام می‌شود. این صفحه به چهار بلوک ۳×۳ تقسیم شده است. بازیکنان به نوبت یک مهره (سفید یا سیاه) در خانه‌های خالی صفحه قرار می‌دهند و سپس یک بلوک ۳×۳ را ۹۰ درجه (ساعتگرد یا پادساعتگرد) می‌چرخانند.

- **هدف:** ایجاد یک خط افقی، عمودی یا مورب متشکل از ۵ مهره از رنگ خودتان.
- **چرخش بلوک:** چرخش بلوک می‌تواند موقعیت مهره‌ها را تغییر دهد و ممکن است باعث ایجاد یا از بین بردن خطوط پنج‌تایی شود.
- **پایان بازی:** اگر یک بازیکن ۵ مهره متوالی بسازد، برنده است. اگر صفحه پر شود و هیچ برنده‌ای وجود نداشته باشد، بازی مساوی است.

پیاده سازی

هدف شما پیاده‌سازی الگوریتم Minimax است. کد بازی به شما داده شده است اما این کد کامل نیست و شما باید بخش های 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 PentagoGame با توجه به آرگومان‌های مد نظر بسازید و پس از پیاده سازی تابع minimax، تابع play را در نمونه‌ی خود صدا زده و نتیجه‌ی بازی به صورت زیر برگردانده می‌شود:

1: بازیکن برنده شده است.

0: بازی مساوی شده است.

1-: کامپیوتر برنده شده است.

همچنین برای استفاده از رابط گرافیکی به منظور پیاده‌سازی راحت‌تر، می‌توانید پرچم ui را هنگام ساخت نمونه True کنید. می‌توانید از تابع print_board نیز برای بررسی صفحه بازی در هر مرحله استفاده کنید.

بررسی نتایج

برای درک کامل الگوریتم کد را با عمق‌های مختلف و بدون هرس 50 الی 100 بار اجرا کنید و میانگین زمان و 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 را نشان دهید.

سوالات

1. آیا میان عمق الگوریتم و پارامترهای حساب شده در بخش بالا روابطی می‌بینید؟ بررسی کنید که عمق الگوریتم چه تاثیراتی بر روی شانس پیروزی، زمان و گره‌های دیده شده می‌گذارد.
2. آیا می‌توان ترتیب دیدن فرزندان هر نود را به گونه‌ای انتخاب کنیم که بیشترین هرس را داشته باشیم؟ اگر جواب شما مثبت است روش خود را توضیح دهید و در غیر اینصورت توضیح دهید که چرا این عمل امکان پذیر نیست.

3. Branching Factor را توضیح دهید و بگویید که با پیشرفت این بازی چه تغییراتی می‌کند؟
4. توضیح دهید که چرا به هنگام هرس کردن الگوریتم بدون از دست دادن دقت خود سریع‌تر می‌شود.
5. چرا در حالاتی که حریف به صورت شانسی عمل می‌کند (مانند این پروژه)، استفاده از minimax بهینه‌ترین روش نیست؟ چه الگوریتمی می‌تواند جایگزین این الگوریتم باشد؟ توضیح دهید.

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