دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

هوش مصنوعی
بهار ۱۴۰۱
استاد: محمدحسین رهبان
گردآورندگان: امین مقراضی، پرهام چاوشیان، متین شجاع

تمرین دوم | **جستجوی محلی، بهینه‌سازی پیوسته** | مهلت ارسال: ۱۸ فروردین

- مهلت ارسال پاسخ تا ساعت ۲۳:۵۹ روز مشخص‌شده است.
- در طول ترم امکان ارسال با تاخیر پاسخ همه‌ی تمارین تا سقف ۷ روز و در مجموع ۲۰ روز، وجود دارد. پس از گذشت این مدت، پاسخ‌های ارسال‌شده پذیرفته نخواهند بود. همچنین، به ازای هر روز تأخیر غیر مجاز ۱۰ درصد از نمره تمرین به صورت ساعتی کسر خواهد شد.
- هم‌کاری و هم‌فکری شما در انجام تمرین مانعی ندارد اما پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد.
- در صورت هم‌فکری و یا استفاده از هر منابع خارج درسی، نام هم‌فکران و آدرس منابع مورد استفاده برای حل سوال مورد نظر را ذکر کنید.
- لطفا تصویری واضح از پاسخ سوالات نظری بارگذاری کنید. در غیر این صورت پاسخ شما تصحیح نخواهد شد.

سوالات نظری (۸۰ نمره)

۱. (۱۰ نمره) گزاره‌های زیر را از نظر درستی یا نادرستی بررسی کنید.

(آ) استفاده از الگوریتم Hill Climbing با $f(s) = h^*(s)$، دستیابی به هدف بهینه را تضمین می‌کند.

(ب) اگر یک مسئله‌ی جستجوی محدود به صورتی داشته باشیم که ضریب انشعاب تمام حالات آن برابر ۱ باشد، الگوریتم Hill Climbing حتما هدف را در صورت وجود پیدا می‌کند.

۲. (۱۰ نمره) به سوالات زیر در مورد الگوریتم تپه‌نوردی پاسخ دهید.

(آ) به نظر شما الگوریتم جستجوی سطح اول [۱] به الگوریتم Hill Climbing برتری دارد؟ چرا؟

(ب) به نظر شما چگونه می‌توان الگوریتم Hill Climbing را برای استفاده در فضای حالت پیوسته بسط داد؟

۳. (۱۰ نمره) در صورتی که در الگوریتم Simulated Annealing هرگز مقدار T را کاهش ندهیم، چه خواهد شد؟

۴. (۱۰ نمره) فرض کنید در حین اجرای الگوریتم Simulated Annealing بر روی یک مسئله‌ی خاص، به یک $iteration$ از اجرای الگوریتم رسیده‌ایم که مقدار دما در آن برابر $T = ۲$ و مقدار تابع هدف برابر ۲۰ است. حال اگر حالت فعلی، ۲ حالت همسایه داشته‌باشد که مقدار تابع هدف برای آن‌ها به ترتیب برابر ۲۵ و ۱۵ باشد، به سوالات زیر پاسخ دهید.

(آ) اگر قصد بیشینه کردن تابع هدف را داشته باشیم، احتمال پذیرش هریک از این دو حالت را در صورتی که به طور تصادفی برای انتخاب به عنوان حالت بعدی انتخاب شده‌باشند، به دست آورید.

(ب) احتمالات قسمت قبل را با هدف کمینه کردن تابع هدف به دست آورید.

۵. (۱۰ نمره) فرض کنید مقدار b در الگوریتم Local Beam Search برابر با k است. روند اجرای این الگوریتم را با اجرای k جست‌وجوی محلی موازی مقایسه کنید.

[۱] BFS

۶. (۱۰ نمره) در الگوریتم ژنتیک ارائه شده در اسلایدهای درس برای مسئله‌ی ۸ وزیر، وجود چند وزیر در یک سطر یا ستون مجاز بود. حال فرض کنید شرط وجود تنها یک وزیر در هر سطر و ستون (در حالات میانی) به مسئله اضافه شده است و با این فرض، مکانیزم $crossover$ و جهش مربوط به الگوریتم ژنتیک را برای مسئله‌ی ۸ وزیرِ جدید، ارائه دهید.

۷. (۱۰ نمره) تحدب مجموعه‌ها و یا توابع زیر را نشان دهید.

(آ)

$$A = \{(x, y) \mid ||x|| \leq y\}$$

(ب)

$$B = \{(x + y) \mid x, y \in B\}$$

با این فرض که B یک مجموعه‌ی محدب است.

(ج)

$$f(x) = x^{۲}y^{۲}, (x, y) \in R^{۲}$$

(د)

$$g(x) = \begin{cases} x, & x > ۰ \\ a, & x = ۰ \\ \infty, & x < ۰ \end{cases}$$

با فرض اینکه a مقداری دلخواه و نامنفی دارد.

۸. (۱۰ نمره) فرض کنید $f : R^{۲} \to R$ یک تابع محدب و مجموعه‌ی A یک زیرمجموعه‌ی محدب از R است. ثابت کنید تابع $g : R \to R$ که به صورت زیر تعریف می‌شود، یک تابع محدب است.

$$g(x) = \min_{y \in A} f(x, y)$$

سوالات عملی (۶۰ نمره)

۱. (۳۰ نمره) در این سوال قصد داریم تا روش‌های بهینه‌سازی را برای توابع مختلف بررسی کنیم. در کنار کدهای نوشته‌شده برای این سوال، باید یک فایل pdf نیز تحویل دهید که شامل توضیحات کدهای نوشته شده باشد.

- تابعی بنویسید که یک تابع تک‌متغیره‌ی دیگر به همراه مقادیر ابتدا و انتهای دامنه‌ی آن را به عنوان ورودی دریافت کند و نمودار این تابع را رسم کند. سپس به کمک نمودار رسم شده برای تابع، بررسی کنید که

(آ) آیا توابع زیر محدب هستند یا خیر؟

(ب) آیا برای توابع غیرمحدب داده‌شده، می‌توان روشی عددی معرفی کرد که به کمک آن روش بتوانیم قرینه‌ی سراسریِ تابع را بیابیم؟ توضیح دهید.

$$\frac{x^{۴}e^{x} - sin(x)}{۲} : [-۲, ۱] \longrightarrow \mathbb{R}$$

$$۵\log(\sin(۵x) + \sqrt{x}) : [۲, ۶] \longrightarrow \mathbb{R}$$
$$\cos(۵\log(x)) - \frac{x^۳}{۱۰} : [۰/۵, ۲] \longrightarrow \mathbb{R}$$

- تابعی بنویسید که یک تابع تک‌متغیره‌ی محدب را دریافت کند و با استفاده از روش کاهش گرادیان، مقدار کمینه‌ی سراسری تابع را محاسبه کند. این تابع باید علاوه بر ورودی گرفتن تابع اولیه، مقدار $learning\ rate$ و بیشینه‌ی دفعات تکرار را دریافت کند. برای یافتن مقدار مشتق تابع در یک نقطه، می‌توانید از تعریف مشتق استفاده کنید. سپس با استفاده از تابعی که نوشته‌اید:

(آ) کمینه‌ی سراسری توابع محدب داده شده در قسمت قبل را به ازای مقادیر زیر برای $learning\ rate$، بدست آورید:

$$[۰/۱, ۰/۴, ۰/۶, ۰/۹]$$

(ب) نتایج به دست آمده را از نظر همگرایی یا واگرایی بررسی کنید و توضیح دهید.

(ج) از این تابع برای یافتن کمینه‌ی سراسری در تابع(های) داده‌شده که نه محدب هستند و نه مقعر، استفاده کنید. مجددا به ازای هرکدام از مقادیر $learning\ rate$ داده‌شده، تابع را ۱۰۰۰ مرتبه اجرا کنید و بگویید در چند درصد مواقع تابع کمینه‌ی سراسری را پیدا می‌کند؟ می‌توانید مقدار واقعی نقطه‌ی کمینه‌ی سراسری را به هر روشی که برایتان ممکن است، بیابید.

- یکی از روش‌های عددی برای یافتن نقطه‌ی صفر یک تابع ($f(x) = ۰$) روش نیوتون‌ـرافسون است. اساس کار این روش به شکل زیر است:

$$x_{i+۱} = x_i - \frac{f(x_i)}{f'(x_i)}$$

همانظور که مشاهده می‌کنید این روش شباهت زیادی به رابطه‌ی کاهش گرادیان دارد؛ با این تفاوت که گویی برای مقدار $learning\ rate$، یک مقدار ثابت را در نظر نمی‌گیرد. از آنجایی که هدف ما یافتن کمینه‌ی محلی ($f'(x) = ۰$) است، رابطه نیوتون به شکل زیر تغییر می‌کند:

$$x_{i+۱} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

تمامی بررسی‌هایی که در قسمت قبل برای روش کاهش گرادیان انجام دادید را برای روش نیوتون‌ـرافسون انجام دهید (برای یافتن مشتق دوم تابع در یک نقطه، از تعریف مشتق دوم استفاده کنید). سپس نتایج را مقایسه و تحلیل کنید.

- حال در این قسمت، باید تابعی همانند تابع قسمت دوم بنویسید؛ با این تفاوت که به جای یک تابع تک‌متغیره، تابعی دومتغیره دریافت کرده و کمینه‌ی سراسری آن را بیابید. برای یافتن مقدار گرادیان تابع در یک نقطه، می‌توانید از تعریف گرادیان استفاده کنید.

با استفاده از تابعی که نوشته‌اید، کمینه‌ی محلی تابع زیر را به ازای مقادیر $[۰/۰۱, ۰/۱, ۰/۱۸, ۰/۲۵]$ برای $learning\ rate$، بیابید:

$$f(x,y) = \frac{۲^x}{۱۰۰۰۰} + \frac{e^y}{۲۰۰۰۰} + x^۲ + ۴y^۲ - ۲x - ۳y$$
$$f : [-۱۵, ۱۵] \times [-۱۵, ۱۵] \longrightarrow \mathbb{R}^۲$$

لازم است که روند الگوریتم را ذخیره کرده و به کمک نمودار نمایش دهید. برای رسم نمودار می‌توانید از تابع داده‌شده در پایان این بخش استفاده کنید. نتایج را گزارش و تحلیل کنید.

```
def draw_points(func, x_1_sequence, x_2_sequence):
    fig = plt.figure(figsize=plt.figaspect(0.5))
    X1, X2 = np.meshgrid(np.linspace(-15.0, 15.0, 1000), np.linspace
    (-15.0, 15.0, 1000))
    Y = func(X1, X2)
    f_sequence = [func(x_1_sequence[i], x_2_sequence[i]) for i in
    range(len(x_1_sequence))]

    # First subplot
    ax = fig.add_subplot(1, 2, 1)

    cp = ax.contour(X1, X2, Y, colors='black', linestyles='dashed',
    linewidths=1)
    ax.clabel(cp, inline=1, fontsize=10)
    cp = ax.contourf(X1, X2, Y, )
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.scatter(x_1_sequence, x_2_sequence, s=10, c="y")

    # Second subplot
    ax = fig.add_subplot(1, 2, 2, projection='3d')

    ax.contour3D(X1, X2, Y, 50, cmap="Blues")
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.scatter3D(x_1_sequence, x_2_sequence, f_sequence, s=10, c="r
    ")

    plt.show()
```

- در این بخش قصد داریم تا با استفاده از الگوریتم $Simulated\ Annealing$ ، روشی برای یافتن نقطه‌ی کمینه در توابع غیرمحدب تک‌متغیره ارائه کنیم.

فرض کنید نقطه‌ی فعلی ما x باشد. همسایه‌های نقطه‌ی فعلی را تمام نقاط عضو بازه‌ی $[x-\alpha, x+\alpha]$ تعریف می‌کنیم. همچنین تابع هزینه را قرینه‌ی مقدار تابع در یک نقطه در نظر می‌گیریم. بدین ترتیب مسئله به یک مسئله‌ی قابل حل با استفاده از الگوریتم $Simulated\ Annealing$ تبدیل می‌شود.

در این بخش باید برنامه‌ای بنویسید که تابع هدف، $stopping\ temperature$، $stopping\ iter$، $alpha$، $initial\ T$ و $gamma$ دریافت کند و نقطه کمینه‌ی سراسری را باز گرداند. منظور از $stopping\ temperature$ پایین‌ترین مقدار دمای مجاز است، یعنی در صورتی که دما از این مقدار کمتر شود، الگوریتم متوقف می‌شود. منظور از $stopping\ iter$ بیشترین تعداد دفعه مجاز تکرار است، یعنی در صورتی که بیشتر از این تعداد اجرا شود، متوقف می‌شود. همچنین منظور از $initial\ T$ مقدار اولیه‌ی دماست.

سپس تابع $schedule$ را به فرم $f(T) = \gamma T$ در نظر بگیرید. از اینجا به بعد منظور ما از تابع هدف، تابعی از بخش اول است که نه محدب است و نه مقعر (در صورت وجود چند تابع، در انتخاب مختارید).

به ازای هریک از مقادیر $[۰/۱, ۰/۲, \ldots, ۱]$ برای مقدار α، تابعی که نوشته‌اید را ۱۰۰۰ مرتبه بر روی تابع هدف اجرا کنید و محاسبه کنید در چند درصد مواقع، کمینه‌ی سراسریِ تابع هدف، درست محاسبه شده‌است. دقت کنید که یافتن مقادیر مناسب برای سایر متغیرهای ورودی به عهده‌ی شماست و باید تلاش کنید مقادیر انتخابی به بهبود عملکرد الگوریتم کمک کنند.

نتایج به دست‌آمده به ازای مقادیر مختلف α را با یکدیگر و همچنین با نتیجه‌ی به دست‌آمده در قسمت دوم برای این تابع، مقایسه کنید.

عملکرد این روش‌ها را تحلیل کرده و در نهایت بگویید که کدام روش را برای بهینه‌سازی چنین توابعی

پیشنهاد می‌دهید.

۲. (۳۰ نمره) در این سوال با استفاده از الگوریتم ژنتیک، بازی زیر را پشت سر بگذارید. این بازی به شکل زیر است و می توانید از این لینک، آن را امتحان کنید.

قوانین بازی:

- بازیکن قرمز باید با خوردن همه goal های بازی یا همان دایره‌های زرد رنگ، از محوطه شروع به محوطه پایان برود (محوطه پایان با مربع سبز رنگ نشان داده شده است).
- بازیکن در هنگام بازی نباید به دشمنان خود که با دایره‌های آبی رنگ مشخص شده‌اند، برخورد کند. در صورت برخورد به آن‌ها می‌بازد.
- بازیکن نمی‌تواند از border بازی عبور کند.

تمرین شما:

- شما باید کد خود را در فایل solve.py بنویسید و نیازی به تغییر فایل بازی ندارید (پیشنهاد می‌شود یک بار محتویات فایل بازی را مطالعه بفرمایید).
- به شما چند فایل txt به اسم‌های map[i].txt داده می‌شود که هرکدام نشان دهنده نقشه‌ای از بازی هستند. نمره هر یک از نقشه‌ها جدا است و کد ژنتیک شما باید هر کدام از آن‌ها را به صورت جداگانه train شود.
- فایل solve.py حاوی ۴ تابع است که طرز کار هر یک از آن‌ها به صورت زیر است:

(آ) با صدا زدن تابع *play_human_mode* که اسم فایل نقشه بازی را به عنوان ورودی می‌گیرد، می‌توانید خودتان بازی را امتحان کنید(بازیکن با کلیدهای wasd حرکت می‌کند).

(ب) تابع *play_game_AI* یک رشته ورودی از حرکات wasd را به عنوان جهت حرکت و نیز x را به عنوان عدم حرکت دریافت می‌کند. سپس آن دنباله از جهات را برای حرکت دنبال کرده و شما نیز می‌توانید آن را مشاهده کنید. سپس بازی انجام شده را که یک object از کلاس Game هست به شما برمی‌گرداند و شما می‌توانید اطلاعاتی را که در پایین به شما داده می‌شود (مانند مختصات نهایی بازیکن) از بازی get کنید.

(ج) تابع simulate مانند تابع بالا است با این تفاوت که گرافیک آن را مشاهده نمی‌کنید و بسیار سریع‌تر انجام می‌شود.

(د) تابع *run_whole_generation* همانطور که از اسم آن مشخص است، برای نشان دادن و تست کردن عملکرد اعضای یک نسل و چندین بازیکن به کار می‌رود و به عنوان ورودی یک لیست از رشته‌های حرکات بازیکنان به همراه طول رشته‌ها دریافت می‌کند (توجه کنید که طول رشته‌ها باید برابر باشند. همچنین می‌توانید به جای رشته از آرایه‌ای از کاراکترها استفاده کنید.).

شما باید یک الگوریتم ژنتیک پیاده‌سازی کنید و برای به دست آوردن fitness جمعیت خود می‌توانید از هرکدام از توابع بالا که می‌خواهید با هر روشی که می‌خواهید آن‌ها را تست کنید و با استفاده از attribute های game که به عنوان خروجی توابع بالا به شما داده می‌شود و در پایین به توضیح آن‌ها می‌پردازیم عملکردی agent های خود را بررسی کنید.

در پایان باید یک گیف برای هر نقشه که عملکرد بهترین agent هر نسل را نشان می‌دهد به همراه فایل solve.py برای ما ارسال کنید.

یک نمودار از امتیازی که بهترین هر نسل گرفته است را در پایان کد خود رسم کنید و screenshot مربوط به آن را نیز با اسم مناسب ارسال کنید.

تابع امتیاز خود را به در گزارشی نیز باید برای ما توضیح دهید که نمودار شما برایمان معنا پیدا کند.

attribute های بازی: پیشنهاد می‌شود علاوه بر اطلاعات زیر، به کد بازی نیز دقت کنید:

- :game.start
این دستور مستطیلی را که در آن شروع به بازی می‌کنید به شما برمی‌گرداند (احتمالا نیازی به آن نخواهید داشت). این مربع یک object از کلاس Start هست که حاوی x, y, h, w است که به ترتیب طول و عرض نقطه ی بالا سمت چپ آن به همراه ارتفاع و عرض خود مستطیل هستند که می‌توانید از آن‌ها استفاده کنید (در پایتون طول و عرض یک نقطه، از بالا سمت چپ screen حساب می‌شوند).
- :game.end
مانند game.start است و نشان دهنده محوطه پایان بازی است.
- :game.Hlines
لیست خط‌های افقی border را برمی‌گرداند.
- :game.Vlines
خط‌های عمودی border بازی را برمی‌گرداند. هر خط در این لیست یک object از کلاس Line هست که attribute های آن $x۱, y۱, x۲, y۲$ هستند که نشان دهنده مختصات شروع و پایان آن پاره‌خط هستند.
- $self.player\_x, self.player\_y$:
مختصاتی که بازیکن در آن شروع به بازی می‌کند را نشان می‌دهد.
- :self.player
در توابعی که یک بازیکن را تست می‌کنند، آن بازیکن را به شما برمی‌گرداند که یک object از کلاس player هست و دارای attribute های $x, y, width, height, vel$ است.
این مشخصات به ترتیب مختصات بازیکن (چون مربع است، مختصات راس بالا سمت چپ آن)، طول و عرض (که در این حالت یکسان هست) و vel یا همان سرعت بازیکن را نشان می‌دهند.
- :self.enemies
دشمنان شما دایره‌های آبی هستند که این دستور یک لیست از آنها را برمی‌گرداند. هر عضو لیست نشان دهنده یک دشمن است که یک object از کلاس Enemy بوده و دارای attribute های x, y, vel, r است که مختصات مرکز و سرعت و شعاع دایره را نشان می‌دهند.
- :self.goals
لیستی از goal های بازی مانند enemies را نشان می‌دهد، با این تفاوت که هر عضو لیست دارای لیست ۲ عضوی دیگر است که عضو اول آن goal و عضو دوم آن خورده شدن یا نشدن goal را نشان می‌دهد.
یک goal نیز مانند یک enemy دارای مختصات و شعاع آن دایره زرد زنگ است.
- :self.hasDied
نشان می‌دهد یک بازیکن زنده است یا خیر.
- :self.hasWon
نشان می‌دهد یک بازیکن برده است یا خیر.

زمانی که تابع $run\_whole\_generation$ صدا زده می‌شود، چون چند بازیکن را با هم تست می‌کنیم، به جای attribute هایی از game که در بالا مشخص کردیم که مربوط به زمان تست یک بازیکن هستند، attribute های دیگر جایگزین شده است که آنهارا ذکر می‌کنیم.

- :self.players
یک لیست از بازیکنان برمی‌گرداند که هر عضو لیست نیز یک لیست ۳ عضوی است که عضو اول آن خود بازیکن است. عضو دوم مبین این است که بازیکن باخته است یا خیر (اگر نبازد ۱ - قرار میگیرد) و عضو سوم نشانگر این است که آیا بازیکن برده است یا خیر.
- $self.goal\_player$:
برای اینکه نشان دهیم آیا i goal ام توسط بازیکن j ام خورده‌شده‌است یا نه استفاده می‌شود و یک جدول ۲ بعدی برمی‌گرداند. به این صورت که اگر عضو i،j آن True باشد، یعنی i goal ام در این run توسط بازیکن j ام خورده شده است.

در صورتی که نیاز به موجودیتی در بازی داشتید که در جدول بالا ذکر نشده است، می‌توانید از دستیار آموزشی مربوطه پرسیده تا option مورد نظر به بازی شما اضافه شود. می‌توانید از هر قابلیتی که در بالا گفته شده نیز استفاده کنید، البته نیاز به استفاده از تمامی آن‌ها نیست.

راهنمایی‌ها:

- چون train کردن کل نقشه زمان‌گیر خواهد بود و در ابتدا حرکات بسیار تصادفی هستند، بهتر است از learning incremental استفاده کنید(با learning incremental در network neural نیز اشتباه نگیرید). learning incremental در ژنتیک به این معناست که به جای اینکه یک ژنوم طولانی بگیرید و به دنبال ژنوم بهینه بگردید، ابتدا یک ژنوم با طول کوتاه‌تر بگیرید و هر چند نسل طول ژنوم را اضافه کنید تا به صورت جزئی از ابتدا تا انتها ژن بهینه پیدا شود.
- برای بهبود نتیجه‌ای که از بازی می‌گیرید می‌توانید از این موارد در تابع امتیازدهی استفاده کنید: فاصله تا مقصد، goal های خورده شده و ...