دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

پروژه سوم

مبانی هوش محاسباتی

(بازی تکامل عصبی)

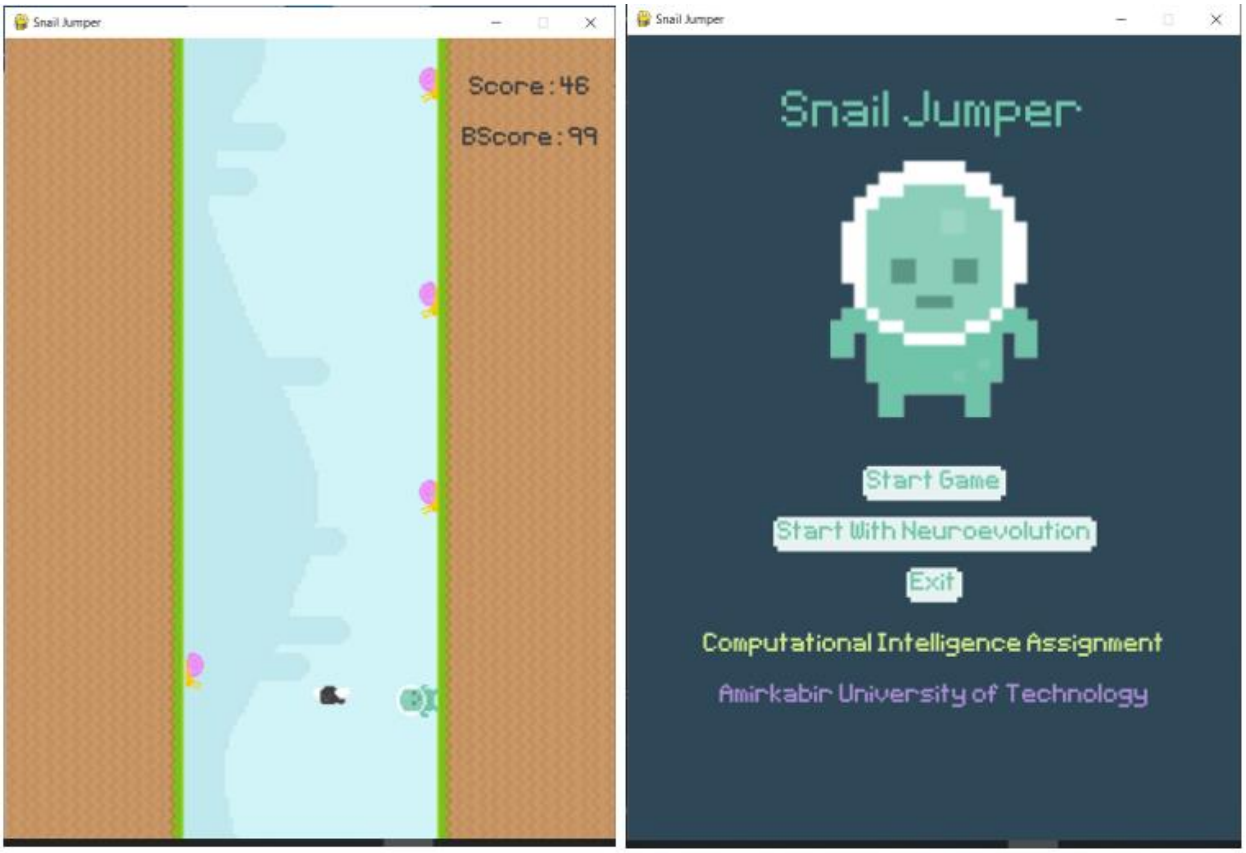
استاد درس: دکتر عبادزاده

زمستان ۱۴۰۰

فهرست

مقدمه و آشنایی با پروژه

در آخرین فصل تدریس شده با الگوریتم‌های تکاملی آشنا شدیم. هدف این پروژه به کاربردن الگوریتم تکاملی برای یادگیری شبکه‌عصبی در محیطی است که داده کافی جهت آموزش آن موجود نمی‌باشد. یکی از این محیط‌ها بازی می‌باشد که همواره اتفاق جدیدی در حال رخ دادن است و لذا تولید داده‌های آموزشی جهت آموزش امری نشدنی است.



شکل ۱- نمای اجرای بازی

بازی طراحی شده این پروژه Snail Jumper نامیده شده‌است. این بازی در دو حالت دستی[1] و تکامل عصبی[2] قابل اجرا می‌باشد. جهت آشنایی با نحوه کار آن، پروژه را از گیت‌هاب دانلود کرده و فایل game.py را اجرا نمایید. پس از اجرا، تصویری مشابه با شکل ۱ تصویر راست مشخص خواهد شد که با انتخاب گزینه اول به صورت

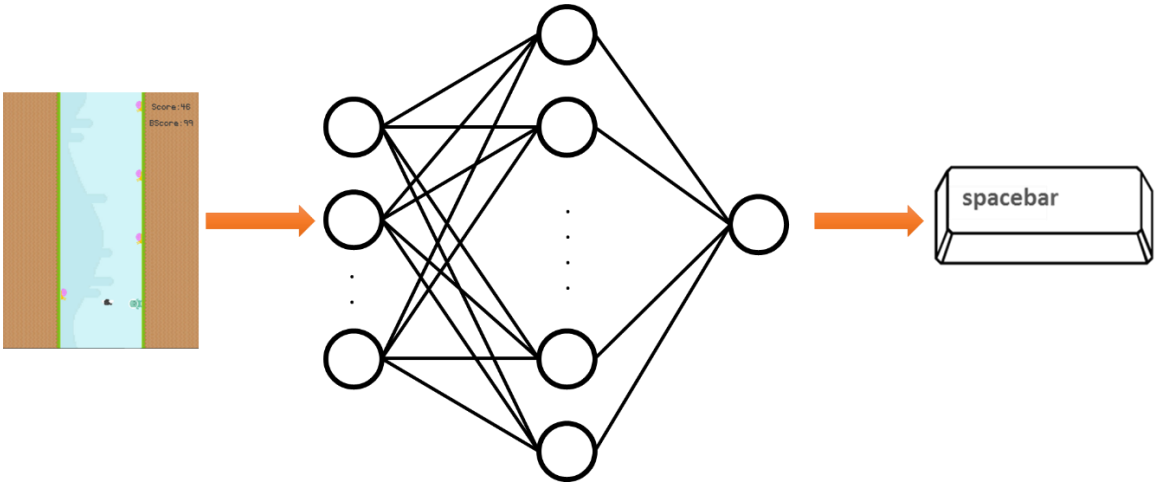
[1] Manual
[2] Neuroevolution

دستی و با انتخاب گزینه دوم به صورت تکامل عصبی می‌توانید آن را اجرا کنید. برای آشنایی، گزینه اول را انتخاب کرده تا بازی اجرا شود. هدف این بازی ساده عبور از موانع سر راه است که با space عمل پرش صورت می‌گیرد.

در ادامه، با نحوه پیاده سازی تکامل عصبی آشنا خواهیم شد و خواهیم دید چگونه الگوریتم تکاملی به یادگیری شبکه عصبی کمک شایانی را خواهند داشت.

شرح مسئله

برای به پیش بردن بازی طراحی شده به صورت تکامل عصبی، بایستی شبکه عصبی‌ای را طراحی کنیم که **پارامترهای مهم در تصمیم گیری** را تحت ورودی در اختیار بگیرد و سپس خروجی متناظر را تولید کند. در انتها خروجی تولید شده به مشابه فشردن دکمه space تعریف شده در بازی عمل کند.



شکل ۲ – نقش شبکه‌عصبی در الگوریتم تکامل عصبی

بنابراین، پس از تعیین پارامترهای مهم در تصمیم‌گیری و ساختن معماری شبکه‌عصبی، عمل Feedforward به‌راحتی صورت می‌گیرد. در فصل شبکه‌عصبی آشنا شدیم که در ادامه کار می‌بایستی تابع هزینه تعریف شود و بعد با Backpropagation، وزن‌ها و بایاس‌ها را به‌گونه ای آپدیت کنیم که به سمت مینیمم شدن میل کند.

اما در مسئله موجود، داده‌ای جهت آموزش و Backpropagation وجود ندارد. لذا در این قسمت از الگوریتم‌های تکاملی می‌توان کمک گرفت. بدین صورت که به تعداد زیاد (در پروژه ما ۳۰۰) بازیکن تولید خواهند شد. هر بازیکن حاوی یک شبکه عصبی می‌باشد که وزن‌ها و بایاس هرکدام به ترتیب رندم نرمال و صفر مقداردهی اولیه شده‌اند. سپس هریک با توجه به معماری شبکه عصبی و مقادیر اولیه موجود، با مشاهده موانع عملکرد متفاوتی را از خود نشان می‌دهند. تعدادی به موانع برخورد کرده و تعدادی نیز عبور خواهند کرد. هرچه بازیکن به مسیر خود بیشتر ادامه دهد، مقدار شایستگی[۳] بیشتری را اختیار خواهدکرد. بنابراین طبق اصل تکامل همواره بازیکن‌های با عملکرد بهتر به نسل بعد منتقل خواهند شد و با در نظر گرفتن عملگرهای تقاطع[۴] و جهش[۵] و با گذر چند نسل، انتظار می‌رود تا عملکرد بهتری را از خود نشان دهند و مسیر بیشتری را طی کنند.

[3] fitness
[4] crossover
[5] mutation

ساختار پروژه

کد پروژه شامل هفت فایل می‌باشد:

- **game.py**: پیاده‌سازی روند اجرای بازی.
- **evolution.py**: حاوی یک کلاس به نام Evolution جهت روند تکامل موجودات هر نسل.
- **nn.py**: معماری شبکه عصبی و بخش Feedforward.
- **player.py**: حاوی کلاس Player برای ساخت بازیکن(ها)ی موجود در صحنه.
- **variables.py**: حاوی متغیرهای عمومی که بین فایل‌ها به اشتراک‌گذاری شده اند.

موارد پروژه

مورد ۱) پیاده‌سازی شبکه عصبی (فایل nn.py):

در **__init__** کلاس، یک لیست پایتون حاوی تعداد نورون هر لایه دریافت خواهد شد. به عنوان مثال درصورت استفاده از [3, 10, 2]، این معنی را می‌دهد که ۳ نورون در لایه ورودی، ۱۰ نورون در لایه پنهان و ۲ نورون در لایه خروجی مورد استفاده قرار خواهد گرفت. شما در این قسمت بایستی با توجه به ورودی های دریافت شده، ماتریس وزن و بردار بایاس های متناظر را ایجاد کنید.

در تابع **activation**، بایستی یک تابع فعالیت نظیر سیگموئید را پیاده سازی کنید.

در تابع **forward** نیز ورودی شبکه عصبی را تحت ورودی تابع در اختیار میگیرید و مسیر feedforward را انجام و نورون(ها)ی لایه آخر را در خروجی برمیگردانید.

مورد ۲) پیاده‌سازی پارامتر های مهم در تصمیم‌گیری مسئله و انتخاب معماری شبکه عصبی (player.py)

در **__init__** کلاس و در خط ۳۸، بایستی معماری مورد استفاده در مسئله را انتخاب کنید. برای مثال لیستی قرار گرفته است که **پاسخ بهینه مساله نمی‌باشد**. لذا بایستی با آزمون و خطا به معماری مناسب‌تری دست‌یابید. توجه شود که یک پاسخ بهینه برای مسئله وجود ندارد و معماری‌های متفاوتی می‌توانند پاسخ بهینه ای را کسب کنند.

در ادامه بایستی به پیاده‌سازی تابع **think** بپردازید. به کمک این تابع ابتدا بایستی با توجه به ورودی های دریافتی تابع، بردار ورودی شبکه عصبی را تشکیل دهید (پیشنهاد می‌شود این عمل در تابع دیگری انجام دهید و در ابتدای این تابع فراخوانی شود). توجه شود که تابع **think** مدام در حین اجرای بازی فراخوانی می‌شود و لذا بایستی پارامترهایی در تصمیم‌گیری انتخاب گردند که به انتخاب پرش به سمت چپ یا راست اثر خواهند گذاشت. پس از تشکیل بردار ورودی، به کمک **self.nn.forward** (تابع پیاده سازی شده در مورد قبل)، خروجی شبکه‌عصبی را تولید کنید و با توجه به خروجی مورد نظر، تابع **self.change_gravity** را فراخوانی کنید.

توجه: همانطور که در کد توضیح داده شده است، در خط ۵۶ تا ۶۰ به صورت تست و نحوه آشنایی با تابع **change_gravity**، کد موقتی قرار داده شده و پس از پیاده سازی شما بایستی این بخش حذف شود.

توجه: همانطورکه در درس آشنا شدید، اندازه مقادیر ورودی در عملکرد شبکه عصبی تاثیر مهمی را خواهند گذاشت. به عنوان مثال اگر نورون های ورودی مقادیر بسیار بزرگی را اختیار کنند، با تغییر وزن و بایاس اگر از تابع فعالیت سیگموئید عبور کنند همواره مقادیر ۱ را اختیار خواهند کرد. لذا یادگیری مناسبی صورت نخواهد گرفت. بنابراین حتما ورودی های مساله را در یک بازه ای نظیر ۰ و ۱ قرار دهید. **برای بخش امتیازی می‌توانید batch normalization نیز در این قسمت مورد نظر قرار دهید (پیاده سازی در nn.py).**

۳) پیاده‌سازی انتخاب بازماندگان (فایل evolution.py):

در صورت اجرای بازی به حالت تکامل عصبی، ۳۰۰ بازیکن ساخته می‌شوند و همگی با توجه به سناریوهای توضیح شده در قبل اجرا می‌شوند. پس از پایان یک نسل (باخت تمامی ۳۰۰ بازیکن)، بخش تکامل به کار می‌آید. هر بازیکن به میزان فاصله که طی می‌کند، مقدار شایستگی آن افزایش می‌یابد. لذا تمامی بازیکن ها یک فیلد با عنوان fitness را اختیار میکنند.

در این بخش به پیاده‌سازی تابع **next_population_selection** می‌پردازیم. در ورودی یک لیستی از شی بازیکنان (که هرکدام یک فیلدی تحت عنوان fitness برای شایستگی دارند) و به علاوه num_players دریافت می‌شود. سپس با توجه به تعداد num_players که در ورودی دریافت می‌شود، بازماندگان برگردانده می‌شوند. برای روش اجباری بایستی به کمک مقدار شایستگی sort شوند و num_players تای اول که بهترین شایستگی دارند انتخاب گردند. **برای بخش امتیازی می‌توانید چرخه‌ی رولت[6] و یا SUS و یا Q تورنومت (هرکدام که عملکرد بهتر را نشان می‌دهد) را در این بخش پیاده سازی کنید.**

توجه: با توجه به فراموش کار بودن روش (μ, λ)، از روش $(\mu + \lambda)$ استفاده شده است. درنتیجه از اجتماع بازیکنان نسل به اتمام رسیده و نسل قبل آن تحت ورودی دریافت شده است.

مورد ۴) پیاده‌سازی انتخاب والدین و تولید موجودات نسل جدید (فایل evolution.py):

پس از انتخاب بازماندگان، حال بایستی والدین انتخاب گردند و به کمک آن ها نسل بعدی (فرزندان) را تشکیل دهیم.

در این بخش بایستی تابع **generate_new_population** پیاده‌سازی شود. در ورودی بازماندگان دریافت میشوند و در خروجی یک آرایه به اندازه num_players از فرزندان برگردانده می‌شوند.

[6] Roulette wheel

در این تابع در صورتی که نسل اول باشد (بازیکنان نسل قبل موجود نباشد)، یک آرایه ای از بازیکنان به صورت رندم برگردانده می‌شود. در غیر این صورت بایستی ابتدا والدین انتخاب شوند و سپس فرزندان تولید شوند. به صورت اجباری، میتوانید تمامی بازماندگان را والد در نظر بگیرید و **برای بخش امتیازی می‌توانید با چرخه‌ی رولت یا SUS یا Q تورنومنت (هرکدام که عملکرد بهتری را اختیار کرد)، والدین را انتخاب کنید**. سپس دو به دو والدان را انتخاب کرده و فرزندان را با عملیات تقاطع و جهش تولید کنید. در انتها لیست فرزندان را بازگردانید.

توجه: فرزندان بایستی آبجکت هایی متفاوت از والدین باشند. لذا برای تولید فرزندان تابع **clone_player** را فراخوانی کنید تا یک نسخه جدید از بازیکن قبلی با همان شایستگی و پارامترهای شبکه عصبی حاصل گردد.

مورد ۵ (امتیازی): منحنی آموزش[۷]

برای تحلیل بهتر فرایند تکامل، در هر نسل، بیشترین، کمترین و متوسط شایستگی بازیکنان را محاسبه کرده و در نهایت پلات کنید. برای این کار، می‌توانید در تابع **next_population_selection**، این کار را انجام دهید. به طوری که در انتهای هر نسل اطلاعات شایستگی آن نسل در یک فایل ذخیره می‌کنید. سپس یک فایل پایتون جدید تعریف کرده و اطلاعات مورد نظر را خوانده و پلات می‌کنید.

[7] Learning Curve

نحوه تحویل و پل ارتباطی

پروژه را به صورت فایل فشرده در سامانه کورسز بارگذاری کنید.

برای پروژه نیازی به **نوشتن گزارش نیست**. و بعد از مهلت انجام، **تحویل آنلاین** خواهیم داشت.

در صورت وجود ابهام یا سوال می‌توانید با تدریسیاران درس با آیدی @SoroushMehraban و @Kasramojallal در تلگرام در ارتباط باشید.