دانشگاه تهران

دانشکده ریاضی آمار و علوم کامپیوتر

(علوم کامپیوتر)

پروژه کارشناسی

عنوان

ساخت و تشخیص ربات‌های انسان‌نما به وسیله شبکه‌های مولد رقابتی

نگارش

پارسا یاهوئی

استاد راهنما

دکتر هدیه ساجدی

نیم سال اول ۱۳۹۹-۱۴۰۰

چکیده

ربات‌های انسان‌نما به ربات‌هایی گفته میشود که توانایی انجام کارهای انسانی را دارند، مانند انسان عمل می‌کنند و در طراحی آن‌ها سعی می‌شود تفاوت‌شان با انسان به حداقل برسد. در فضای دیجیتال، رو به رو شدن با چنین بات‌هایی به دو شکل است، گاهی صاحب یک محصول می‌خواهد از انسان بودن مخاطبش مطمئن باشد و عده‌ای برای سواستفاده یا برخی دلایل دیگر میخواهند سیستم را فریب دهند و رباتی را جایگزین انسان کنند. گاهی نیز صاحب یک محصول برای کاهش هزینه نیروی انسانی و یا دلایل دیگر، میخواهد از ربات استفاده کند و شبه انسانی بودن این ربات، باعث بهبود کیفیت تجربه کاربر می‌شود که این موضوع، مطلوب صاحب محصول است.

در صنعت بازی‌سازی نیز از هر دو شکل ذکر شده استفاده می‌شود. شکل اول در تشخیص آنکه بازیکن متقلب نیست و یا رباتی به جای خود نگذاشته است و شکل دوم برای آموزش کاربر و یا ساخت تجربه مناسب در اوایل انتشار بازی که تعداد بازی‌کنان بسیار کم هستند.

روشی که در این پروژه استفاده شده، ساخت شبکه‌هایی ناسازگار است که بتوانند هر دوی این مشکلات را برطرف کنند. مراحل آموزش شبکه با استفاده از اطلاعات جمع آوری شده در حین انجام پروژه انجام می‌شود.

کلمات کلیدی:

ربات انسان‌نما، شبکه عصبی، شبکه مولد ناسازگار، یادگیری تقویتی، تشخیص بات

فهرست مطالب

فصل ۱

مقدمه

۱.۱ ربات انسان‌نما چیست؟

این ربات‌ها به منظور شبیه‌سازی رفتار انسان ساخته می‌شوند. طوری که در صورت تعامل با آن‌ها متوجه انسان نبودن آن‌ها نشوید. بسیاری سعی دارند این گونه ربات‌ها را از لحاظ فیزیکی شبیه به انسان بسازند تا یک تجربه کامل انسانی را فراهم کنند، اما این مسئله محدود به ربات‌هایی است که قطعات فیزیکی دارند و قرار است کاری فیزیکی انجام دهند. پیچیدگی این نوع ربات‌ها بسیار زیاد است زیرا باید احساسات، صدای انسانی، میمیک صورت و بسیاری چیزهای دیگر را شبیه‌سازی کنند.

در فضای دیجیتالی مسئله کمی متفاوت است. روابط و شکل ارتباط گیری محدود به ابزارها می‌شود. مثلا در یک پیام‌رسان دیگر چهره‌ای وجود ندارد که تقلیدش پیچیدگی زیادی داشته باشد و یا صدایی نیست. از این رو ربات‌های موجود روی این بسترها نیز با توجه به محدودیت‌های آن‌ها فعالیت خود را انجام می‌دهند. محدودیت‌هایی که که باعث سهولت در ساخت این ربات‌ها می‌شود.

یکی از کارهای جالبی که توسط آلن تورینگ[۱]، ریاضیدان مشهور، انجام شده در همین باره است. آزمایش تورینگ که یکی از منابع الهام این پروژه است، روشی برای سنجش میزان هوشمندی ماشین است. آزمون به این صورت انجام می‌گیرد که یک شخص به عنوان آزمایشگر، با یک ماشین و یک انسان به گفتگو می‌نشیند، و سعی در تشخیص ماشین از انسان دارد. در صورتی که ماشین بتواند قاضی را به گونه‌ای بفریبد که در قضاوت خود دچار اشتباه شود، توانسته‌است آزمون را با موفقیت پشت سر بگذارد.

۲.۱ ربات‌ها در بازی‌های رایانه‌ای

هدف از مورد استفاده قرار دادن یک ربات در بازی رایانه‌ای، سهولت امر طراحی آن است. گاهی برای ساخت یک محیط پویا در بازی لازم است ماه‌ها وقت گذاشته شود و تیم‌های طراحی، فنی و... با هم درگیر باشند تا بتوانند حالت‌های مختلف را در نظر بگیرند و برای هر مورد پاسخی درخور فراهم کنند، در این صورت ممکن است مواردی از قلم بیفتد و حین بازی کردن، بازیکن با موقعیتی روبه‌رو شود که ادامه فعالیت را غیر ممکن میکند. به همین دلیل استفاده از توان کامپیوتر، امری بدیهی به نظر می‌آید.

ربات‌های مورد استفاده این صنعت به دو بخش عمده تقسیم می‌شوند: ربات‌هایی که در حین ساخت به کمک

---

[۱] *Alan Mathison Turing*

سازنده‌ها می‌آیند، مانند ربات‌هایی برای ایجاد تعادل در بازی و ربات‌هایی که پس از ساخت و هنگام استفاده کاربر به کار می‌آیند، مانند ربات‌های بازیکن و ربات‌های تنظیم کننده درجه سختی یک بازی به صورت بلادرنگ.

برخی ربات‌ها کارکردی فراتر از یک انسان دارند، مانند رباتی که وظیفه تولید محتوا بر اساس تصمیمات بازیکن را بر عهده دارد. انسان‌نما بودن این ربات‌ها موضوعیتی ندارد و در این پروژه به بررسی این موارد پرداخته نمی‌شود.

ربات‌های انسان‌نما که برای رقابت با بازیکن و یا همراهی و همکاری با او طراحی می‌شوند، از چالش برانگیزترین انواع این ربات‌ها هستند. چرا که باید از یک طرف کاملا شبیه انسان رفتار کنند و از طرف دیگر رفتاری منطقی بر اساس قوانین بازی داشته باشند. این ربات‌ها تا به حال با کمک ناظر بیرونی طراحی می‌شدند و مراحل یادگیری تنها در جنبه قوانین بازی اعمال می‌شد. حتی گاهی برای متعادل کردن رفتار ربات، محدودیت‌هایی بر روی سرعت اعمال تصمیم میگذارند تا رفتاری شبیه‌تر به انسان داشته باشد.

۳.۱ هدف پروژه

هدف نهایی این پروژه در مرحله اول ساخت رباتی انسان‌نما بدون کمک ناظر بیرونی است. رباتی که از نحوه بازی کردن بازیکنان عادی، رفتار انسانی را بیاموزد و آن را تقلید کند.

برای مثال اگر مدل با اطلاعات افراد مبتدی آموزش داده شده باشد، در حین بازی حرکاتی شبیه به آنان انجام داده و احتمال باختش نیز بیشتر است. در این صورت نمی‌توان گفت که مدل بد عمل می‌کند بلکه نتیجه نهایی آن باید به شکل کاملا توصیفی مورد بررسی قرار گیرد. در واقع رفتار انسان‌گونه و غیر قابل تمییز ملاک عملکرد است.

۴.۱ چگونگی ساخت ربات

فرایند آموزش در این مدل به معنی آموزش دو مدل، یکی مولد و دیگری تشخیص دهنده است و پس از اتمام فرایند، از مدل مولد به عنوان ربات انسان‌نما و از مدل تشخیص دهنده نیز به عنوان یک ناظر استفاده می‌شود.

اطلاعات بازیکنان به عنوان مجموعه‌ای از اعمال و موقعیت‌ها استخراج می‌شود. خروجی مولد نیز به همین صورت خواهد بود.

از آنجا که شبکه مولد قرار است به صورت بلادرنگ عمل کند، انتظار می‌رود به ازای موقعیت یک عمل را انتخاب کند، پس برای ساخت خروجی بخش مولد، شبکه را در بازی قرار می‌دهیم و مجموعه عمل و موقعیت آن را استخراج میکنیم و به عنوان ورودی در اختیار شبکه تشخیص دهنده میگذاریم.

با توجه به شرایط برای ساخت شبکه مولد از شبکه عمیق کیو [۲] استفاده شده و برای شبکه تشخیص دهنده نیز از یک شبکه عصبی پیشخور [۳] استفاده شده است.

[۲] *Deep Q-Network*
[۳] *Feed Forward Neural Network*

فصل ۲

پیش نیازها

اولین قدم برای شناخت هر مبحث جدید آشنایی با مفاهیم اولیه آن است. در این پروژه با مفاهیم و مدل‌های متفاوتی روبه‌رو هستیم که نیاز است آن‌ها را مطالعه و بررسی کنیم. از طرفی عدم وجود مجموعه دادگان مناسب ضرورت جمع آوری چنین دادگانی را ایجاب می‌کند که درباره نحوه جمع آوری دادگان نیز توضیحاتی ارائه شده است.

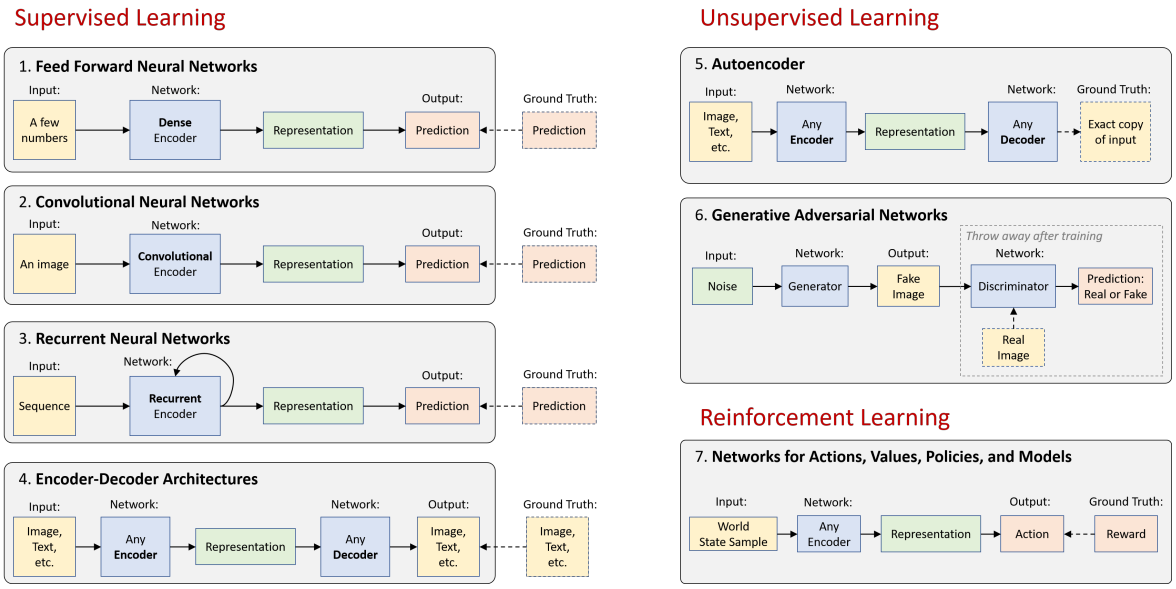
۱.۲ بررسی شبکه‌های عصبی

ابتدا به بررسی برخی اصطلاحات و تعاریف اولیه می‌پردازیم. یادگیری عمیق [۱] یک یادگیری الگویی [۲] در بین داده‌های موجود است. یادگیری الگویی یک نوع یادگیری است که به نمایش داده های ورودی می‌پردازد، و در این مسیر معمولا از تبدیل‌های خطی و غیرخطی یا استخراج ویژگی‌هایی از داده‌ها استفاده می‌کند. این مسیر انجام کار را برای مسائلی نظیر طبقه بندی [۳] یا پیش بینی [۴] آسان تر می کند. یادگیری عمیق زیر مجموعه شاخه گسترده‌تری به نام یادگیری ماشین [۵] است. یادگیری ماشین یک مطالعه علمی از الگوریتم‌ها و مدل‌های آماری است که سیستم‌های رایانه‌ای برای انجام یک کار خاص بدون استفاده از دستورالعمل صریح استفاده می کنند. روش های مختلفی برای یادگیری نمایش های مختلف وجود دارد که موضوع بحث ما نیست. با توجه به تعاریف بالا یادگیری عمیق یک نوع یادگیری الگویی است که به صورت خودکار تلاش به ساخت الگوهای پنهان در داده ها میکند. در این مسیر نحوه نمایش داده ها میتواند مدل را برای انسان‌ها و ماشین‌ها ساده‌تر کند.

تصویر ۱.۲ بخش بندی کلی برای شبکه‌های عصبی در این حوزه‌هاست. موارد ۱-۶ برای مسائلی از قبیل اطلاعات بصری (تصویر و ویدیو)، زبان (متن) و صوت به‌کار می‌روند. مورد ۷ بر اساس سیستم (یا قوانین دنیای معمولی) با روش جزا و پاداش کار میکند.

---

[۱] *Deep Learning*
[۲] *Representation Learning*
[۳] *Classification*
[۴] *Prediction*
[۵] *Machine learning*



شکل ۱.۲: بخش‌بندی مهم‌ترین دسته‌های شبکه‌های عصبی به همراه ساختار توصیفی لایه‌های آنها، برگرفته از ***https://www.edxcope.com/wp-content/uploads/2020/07/DAY-02-Lecture.pdf***

۲.۲ شبکه‌های عصبی پیشخور

شبکه عصبی پیشخور [۶] یک شبکه عصبی مصنوعی است، که در آن اتصال میان واحدهای تشکیل دهنده آن یک چرخه را تشکیل نمی‌دهند. شبکه عصبی پیشخور اولین و ساده‌ترین نوع شبکه عصبی مصنوعی می‌باشد. در این شبکه اطلاعات تنها از یک مسیر حرکت می‌کند که جهت آن رو به جلو می‌باشد. در واقع اطلاعات با شروع از گره (نورون)های ورودی و گذر [۷] از لایه‌های پنهان (در صورت وجود) به سمت گره‌های خروجی می‌روند. همان‌طور که گفته شد در این شبکه حلقه یا دوری وجود ندارد. از نظر فنی، بیشتر شبکه‌ها در یادگیری عمیق می توانند پیشخور در نظر گرفته شوند، اما معمولاً به ساده ترین نوع آن‌ها برای مثال پرسپترون چند لایه متراکم متصل [۸] اشاره دارد. موارد استفاده این شبکه در طبقه بندی برای مسائل گسسته یا رگرسیون برای مسائل پیوسته است.

۳.۲ شبکه‌های عصبی پیچشی

شبکه‌های پیچشی [۹] که با نام دیگر ***convNet*** نیز شناخته می‌شوند، شبکه‌های پیشخوری هستند که با استفاده از تکنیک ثبات مکانی [۱۰] به صورت کارآمدتری به یادگیری الگو و طرح‌های محلی می‌پردازند. این شبکه‌ها غالباً در داده‌های تصویری استفاده می‌گردند.

ثبات مکانی یا تغییر ناپذیری مکانی به این معنی است که، یک گوش گربه در گوشه بالای تصویر همان ویژگی‌هایی را داراست که گوش گربه در گوشه پایین تصویر از آنها برخوردار است. شبکه‌های پیچشی، وزن را در فضا به اشتراک می گذارند تا تشخیص گوش گربه‌ها و سایر الگوها با هزینه محاسباتی کمتر و کیفیت بهتری صورت پذیرد. آنها

[۶] *Feed Forward Neural Network*
[۷] *pass*
[۸] *Multi Layer Perceptron (MLP)*
[۹] *Convolutional Neural Network*
[۱۰] *Spatial in-variance*

بجای اینکه تنها از لایه‌های متراکم [۱۱] استفاده کنند، از لایه‌های پیچشی [۱۲] نیز استفاده می‌کنند. این شبکه ها برای طبقه بندی تصویر، تشخیص اشیاء، تشخیص فعالیت در ویدیو و هرگونه داده‌ای که در ساختار آن نوعی ثبات مکانی موجود باشد، استفاده می شوند. این نوع شبکه‌ها در مبحث بهینه سازی فیلم برای تشخیص اشیای تکراری نیز استفاده می‌گردند؛ به این صورت که اگر دست خود را از سمت راست به سمت چپ حرکت دهید تصویر دست شناسایی شده و در فریم‌های بعدی کپی میشود. در این پروژه برای مدل مولد و تشخیص دهنده نیز از شبکه های عمیق پیچشی استفاده شده است.

۴.۲ شبکه‌های عصبی بازگشتی (بازخوردی)

شبکه‌های عصبی بازگشتی [۱۳] جزء دسته‌ای از شبکه‌های عصبی هستند، که حلقه (چرخه) دارند که به "حافظه" معروف است و این حافظه‌ی ذخیره‌سازی می‌تواند تحت کنترل مستقیم شبکه‌ی عصبی باشد. اتصالات بین گره‌ها از یک گراف جهت‌دار در این شبکه‌ها در امتداد یک دنباله‌ی زمانی می‌باشند و سبب می‌شود تا شبکه بتواند رفتار پویای موقتی را به نمایش بگذارد. برخلاف شبکه‌های عصبی پیشخور، شبکه‌های عصبی بازگشتی می‌توانند از حافظه (وضعیت درونی) خود برای پردازش دنباله‌ی ورودی‌ها استفاده کنند، که آن‌ها را برای مواردی نظیر تشخیص صوت، یا تشخیص دست‌نوشته‌های غیربخش‌بندی شده، مناسب می‌کند. درست همانطور که شبکه‌های پیچشی نیز وزن را در فضا به اشتراک می‌گذارند، شبکه‌های بازگشتی نیز وزن‌ها را به اشتراک می‌گذارند که به آنها اجازه می‌دهد تا الگوهای داده‌های متوالی را به صورت کارا پردازش کنند. انواع مختلفی از ماژول های شبکه‌های بازگشتی وجود دارند، برای مثال شبکه‌های ***LSTM*** و ***GRU***، که برای کمک به یادگیری الگوها در توالی‌های طولانی ساخته شده‌اند.

۵.۲ شبکه‌های عصبی خود کدگذار (خود رمزگذار)

شبکه‌های خود کدگذار [۱۴] یکی از ساده‌ترین اشکال یادگیری بدون نظارت [۱۵] هستند که برای ساختار معماری خود از ساختار ***encoder-decoder*** استفاده می‌کند و سعی می‌کند یک کدینگ دقیق از داده‌های ورودی تولید کند. در ساده ترین حالت یک خود کدگذار شامل یک رمزگذار [۱۶] و رمزگشا[۱۷] به همراه تنها یک لایه پنهان می‌باشد. ورودی به رمزگذار داده شده و خروجی از رمزگشا استخراج می‌شود. در این نوع شبکه به جای آموزش شبکه و پیش‌بینی مقدار تابع هدف در ازای ورودی X، خود رمزگذار آموزش می‌بیند که ورودی خود را بازسازی کند؛ بنابراین بردار خروجی همان ابعاد بردار ورودی X را خواهد داشت؛ یعنی تعداد نورونهای موجود در لایه ورودی و خروجی با یکدیگر برابر است.از آنجا که نمایش لایه‌ی میانی بسیار کوچکتر از داده های ورودی است، شبکه مجبور است یاد بگیرد که چگونه معنادارترین نمایش را تشکیل دهد. خود کدگذارها با حداقل کردن خطای بازسازی شبکه را آموزش میدهند. معمولا تعداد نورونهای موجود در لایه پنهان کمتر از لایه ورودی/خروجی می‌باشد. لایه پنهان در حقیقت باز نمایشی از داده در فضا با بعد کاهش یافته می‌باشد و به معنی ویژگی‌های استخراج شده است. پس از آموزش شبکه بخش رمزگشا حذف شده و خروجی میانی ترین لایه پنهان به عنوان ویژگی های استخراج شده در نظر گرفته میشود. از کاربردهای این شبکه می‌توان به تغییر شکل تصویر، استخراج ویژگی و فشرده سازی نمایش

Fully Connected (Dense) [۱۱]
convolutional layers [۱۲]
Recurrent Neural Network [۱۳]
Auto Encoder [۱۴]
Unsupervised learning [۱۵]
encoder [۱۶]
decoder [۱۷]

داده‌های با ابعاد بالا، یا به عبارت دیگر برای کاهش ابعاد اشاره کرد.

۶.۲ شبکه های عصبی مولد ناسازگار

شبکه‌های مولد ناسازگار [۱۸]چارچوبی [۱۹] برای دیگر شبکه‌هاست که در آن به کمک بهینه‌سازی چند متغیره و داده‌های مسئله تلاش می‌شود، نمونه‌های جدید و دیده نشده را تولید کرد. در ساده‌ترین شکل، روند آموزش شامل دو شبکه است. یک شبکه به نام مولد، نمونه‌های جدیدِ داده را تولید می کند، و سعی می‌کند شبکه دیگر یعنی تشخیص‌دهنده را که نمونه‌های واقعی را از جعلی تشخیص می‌دهد، فریب دهد. اگرچه این شبکه‌ها در ابتدا به عنوان نوعی الگوی تولید برای یادگیری بدون نظارت پیشنهاد شده بودند، اما آنها برای یادگیری نیمه نظارت[۲۰]، یادگیری کاملاً نظارت شده [۲۱]، و یادگیری تقویت کننده [۲۲] مفید هستند.

طی چند سال گذشته، انواع متفاوتی از این دسته شبکه‌ها ارائه شده و این شبکه‌ها پیشرفت‌ها چشمگیر و کاربردهای فراوانی را کسب کرده‌اند، از جمله توانایی آنها می‌توان به تولید تصاویر از یک کلاس خاص، امکان نقشه‌برداری تصاویر و افزایش وضوح اشاره کرد. همچنین کاربردهای جدید این شبکه‌ها در صنعت‌های مد و فشن، بازی‌سازی و ابزارهای تشخیص هویت نیز در حال گسترش است. همچنین در مباحث پزشکی نیز فعالیت‌هایی نظیر شناسایی گلوکوماتوز [۲۳] از روی تصاویر برای تشخیص زودهنگام و جلوگیری از بین رفتن جزئی یا کلی بینایی، انجام شده است.

۷.۲ یادگیری تقویتی

یادگیری تقویتی [۲۴] نوع مهمی از یادگیری ماشین است که در آن یک عامل می‌آموزد چگونه در محیط با انجام اقدامات و دیدن نتایج آن‌ها رفتار کند. در سال‌های اخیر، بهبودهای متعددی در این حوزه پژوهشی فوق‌العاده به وقوع پیوسته است. دیپ‌مایند [۲۵] و معماری یادگیری Q عمیق [۲۶] در سال ۲۰۱۴، غلبه بر قهرمان بازی گو ***GO*** با آلفاگو [۲۷] در سال ۲۰۱۶ و ***OpenAI*** و ***PPO*** در سال ۲۰۱۷ تنها برخی از این پیشرفت‌ها هستند.

در این نوع یادگیری، عامل با رسد کردن محیط و تشخیص وضعیت آن یک عمل را انتخاب می‌کند و تغییرات محیط را همراه با پاداش و یا جزا دریافت می‌کند. تلاش این عامل در راستای رسیدن به هدف و دریافت حداکثری پاداش و حداقلی جزا می‌باشد.

۸.۲ یادگیری تقویتی عمیق

تفاوت یادگیری تقویتی عمیق با یادگیری تقویتی، در شبکه عصبی استفاده شده آن است. برای محاسبه مقدار تابع Q در یادگیری تقویتی، بسته به فضای حاصل از ضرب مجموعه اعمال و موقعیت‌ها می‌توان روش‌های متفاوتی

---

[۱۸] ***Generative Adversarial Networks***
[۱۹] ***Frameworks***
[۲۰] ***semi-supervised learning***
[۲۱] ***fully-supervised learning***
[۲۲] ***reinforcement learning***
[۲۳] ***glaucomatous***
[۲۴] ***Reinforcement Learning, RL***
[۲۵] ***DeepMind***
[۲۶] ***Deep Q learning***
[۲۷] ***AlphaGo***

را برگزید. اگر این مجموعه آنقدر بزرگ نبود، می‌توان از جدول Q استفاده کرد، اما در صورت بزرگ بودن این مجموعه و یا پیوسته بودن آن، یکی از راه‌های پیشنهادی استفاده از شبکه‌های عصبی است. در صورت استفاده از شبکه عصبی، با توجه به عمیق بودن یا نبودن این شبکه، نامگذاری اتفاق می‌افتد.

۹.۲ Q - شبکه عمیق

استفاده از شبکه عصبی عمیق در یادگیری تقویتی باعث تولید شبکه عصبی منحصر به فردی می‌شود که با چهار تکنیک سعی می‌شود از ناپایدار بودن روند یادگیری جلوگیری شود:

- ***Experience Replay***
- ***Target Network***
- ***Clipping Rewards***
- ***Skipping Frames***

تکنیک ***Experience Replay*** : شبکه عصبی عمیق در اپیزودها به راحتی دچار بیش برازش می‌شود، پس از این اتفاق ساخت تجربه‌های متفاوت سخت می‌شود. از این رو تجربه را که حاوی موقعیت‌ها، اعمال و پاداش‌هاست را ذخیره می‌کنند تا بتوان از آن‌ها استفاده کرد.

تکنیک ***Target Network*** : تغییرات تابع هدف بسیار زیاد است و این ناپایداری فرایند یادگیری را سخت می‌کند. برای همین منظور در این تکنیک متغیرهای تابع هدف را ثابت فرض میکنند و هر ۱۰۰۰ قدم، آخرین مقادیر شبکه را جایگزین می‌کند.

تکنیک ***Clipping Rewards*** : در بازی‌های متفاوت میزان پاداش و جزا متفاوت است، مثلا در بازی شطرنج مهره‌های متفاوت امتیازهای متفاوتی نیز دارند، اما در بازی پینگ پنگ در صورت باخت تنها یک امتیاز منفی گرفته می‌شود. برای رفع این مشکل، در این تکنیک همه پاداش‌ها را برابر +۱ و همه جزاها را برابر -۱ در نظر میگیرند.

تکنیک ***Skipping Frames*** : بازی‌ها با سرعت ۶۰ فریم بر ثانیه تغییر می‌کنند، اما عمل بازیکنان به این سرعت نیست، عامل نیز نیازی به پاسخدهی در هر فریم ندارد. در این تکنیک محاسبه تابع Q را هر ۴ فریم انجام می‌دهند و به عنوان ورودی شبکه ۴ فریم پشت سر هم را در اختیار می‌گذارند.

۱۰.۲ انتخاب مجموعه دادگان

یکی دیگر از مواردی که در ابتدای کار باید مشخص گردد، انتخاب پایگاه دادگان مناسب برای مسئله مورد بررسی است. انتخاب مجموعه دادگان کار بسیار دشواری است، از آن جهت که نه تنها باید به موضوع پژوهش و ارتباط آن با دادگان توجه کرد بلکه باید به توان محاسباتی مورد نیاز در مرحله آموزش نیز توجه کرد. انتخاب مجموعه دادگان بزرگ و بیشتر از توان محاسباتی و یا مجموعه دادگان بسیار کوچک صحیح نمی‌باشند زیرا ممکن است، منجر به تولید مشکل در فرایند آموزش شوند و یادگیری مدل به درستی انجام نشود.

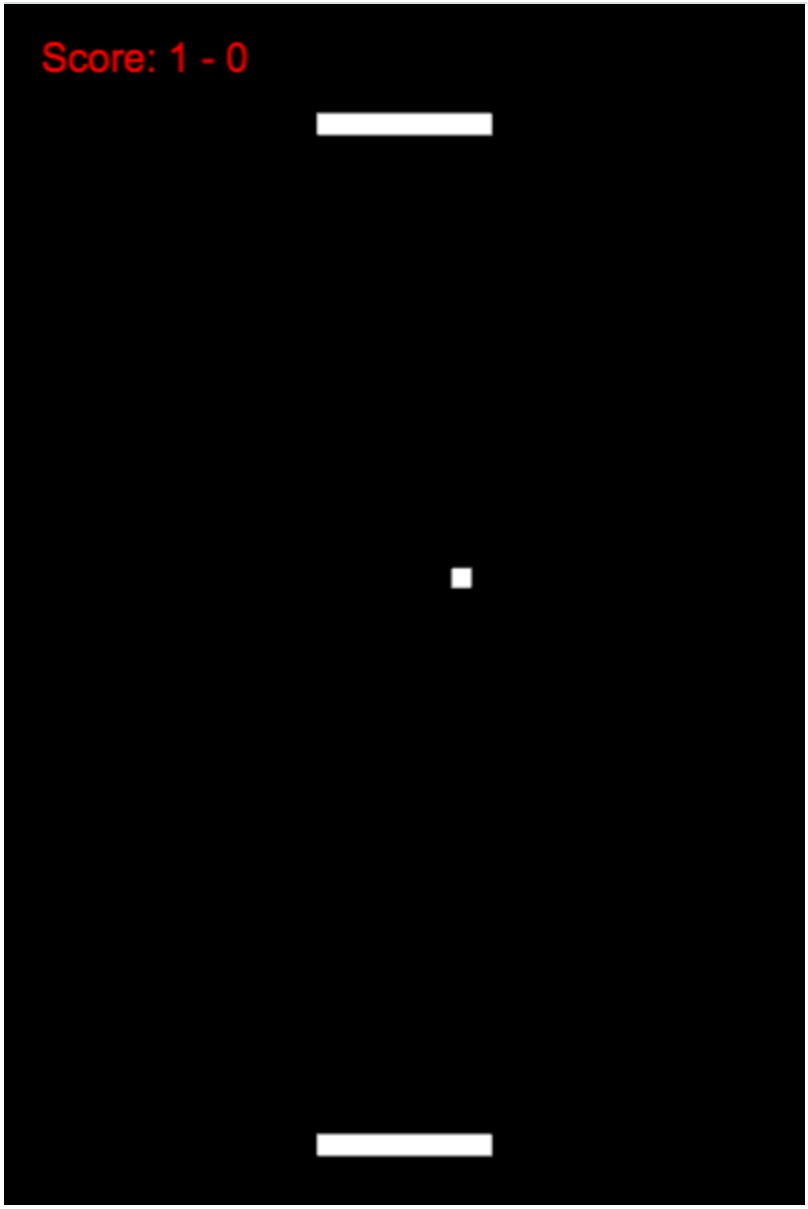
محیط‌های پویای زیادی از جمله ***ALE*** وجود دارد که بستر یادگیری تقویتی را فراهم می‌کنند، اما هیچ یک دادگانی از نحوه بازی کردن انسان‌ها ندارند. از طرفی بازی‌های موجود در این مجموعه‌ها نسخه قابل استفاده برای انسان‌ها نداشت. مشکل دیگری که وجود داشت، خروجی این بازی‌ها به صورت فریم بازی بودند که کار استخراج

دادگان انسانی را سخت می‌کرد. به همین خاطر برای انجام این پروژه تصمیم بر آن شد که یک بازی ساده به نام ***pong*** ساخته شود تا هم برای آموزش شبکه عصبی و هم برای جمع آوری دادگان انسانی مورد استفاده قرار گیرد. تصویر آن در شکل ۲.۲ آمده است.

برای ذخیره دادگان انسانی، بازی در اختیار افراد قرار گرفت و در حین بازی کردن آن‌ها هر ثانیه ۱۰ عمل ذخیره می‌شود. دقت بیش از این باعث افزایش حجم داده می‌شود در حالی که سرعت عمل بازی‌کنان کمتر از این است.

در نهایت مجموعه عمل‌ها به صورت رشته‌ای از مشخصه‌های ***R: Right*** و ***L: Left*** است. و مجموعه موقعیت‌ها آرایه‌ای از مختصات مکانی توپ و سکو.

پاداش در صورت برد، +۱ امتیاز و جزا در صورت باخت -۱ امتیاز خواهد بود.



شکل ۲.۲: تصویری از بازی

فصل ۳

چگونگی کارکرد شبکه‌های مولد ناسازگار

شبکه‌های مولد ناسازگار از دو مدل تشکیل شده‌اند، یکی مدل تولیدی (مولد) G [۱] و دیگری مدل تشخیص‌دهنده D [۲]. مدل تولیدی G را می‌توان به عنوان جاعل در نظر گرفت، مدلی که سعی در ایجاد چیزی جعلی، ارائه آن به عنوان اصل و فریب دادن سیستم را دارد. در حالی که مدل تشخیص دهنده D، مشابه پلیس در تلاش برای تشخیص اصل و جعل است. این رقابت آنقدر ادامه می یابد تا مدل G به اندازه کافی هوشمند شود که مدل تشخیص دهنده را با موفقیت فریب دهد. به عبارت دیگر، نقش تشخیص‌دهنده آن است که بین داده‌های واقعی و تولیدشده فرق بگذارد و نقش تولیدکننده اینکه تا حد امکان داده‌های واقعی‌تری را تولید کند.

۱.۳ نماد گذاری

در ادامه از نمادهای زیر برای انسجام متن استفاده خواهد شد.

$D =$ ***Discriminator***

$G =$ ***Generator***

$\theta_d =$ ***Parameters of discriminators***

$\theta_g =$ ***Parameters of generator***

$P_z(z) =$ ***Input noise distribution***

$p_{data}(x) =$ ***Original data distribution***

$p_g(x) =$ ***Generated distribution***

۲.۳ محاسبه تابع هزینه

آنتروپی متقاطع [۳] بین دو توزیع احتمال p، q روی یک مجموعه داده شده، به صورت زیر تعریف می شود.

$$H(p,q) = E_p[-\log q] = H(p) + D_{KL}(p||q)$$

که در آن $H(p)$ آنتروپی p و $D_{KL}(p||q)$ کی.ال دیورژانس [۴] از p به q است. همچنین به $D_{KL}(p||q)$ آنتروپی

---

[۱] *Generator*
[۲] *Discriminator*
[۳] *Cross entropy*
[۴] *Kullback–Leibler divergence*

نسبی [۵] از q با در نظر گرفتن توزیع p نیز می‌گویند و همچنین $E_p[.]$ امید ریاضی با در نظر گرفتن توزیع p است.
برای توزیع‌های احتمالی گسسته p و q داریم:

$$H(p,q) = -\sum_{x\in\chi} p(x)\log q(x)$$
$$D_{KL}(P||Q) = \sum_{x\in\chi} P(x)\log(\frac{P(x)}{Q(x)})$$

۳.۳ ارتباط آنتروپی با لگاریتم تابع درست‌نمایی

می‌خواهیم در دسته مسائل طبقه‌بندی و دسته‌بندی، احتمال نتایج مختلف را تخمین بزنیم. اگر احتمال برآورد شده از نتیجه i برابر q_i و فراوانی آن در داده‌های آموزشی [۶] برابر p_i باشد و دادگان آموزشی شامل N داده باشند، آنگاه داریم که درست نمایی مجموعه داده‌های آموزشی متناسب است با $\Pi_{qi}^{N_{pi}}$. بنابراین، لگاریتم درست‌نمایی تقسیم بر N برابر است با

$$\frac{1}{N}\log\Pi_{qi}^{N_{pi}} = \sum_i p_i \log q_i = -H(p,q)$$

پس میتوانیم بگوییم که ماکزیمم کردن درست‌نمایی معادل مینیمم کردن آنتروپی متقاطع است.

۴.۳ مینیمم کردن آنتروپی متقاطع

مینیمم کردن آنتروپی متقاطع در بهینه سازی اغلب استفاده می شود. هنگام مقایسه یک توزیع در برابر توزیع مرجع (مورد تحقیق) ثابت p در آنتروپی متقاطع و واگرایی در KL با اختلاف یک عدد ثابت مانند c با هم برابرند. هر دو مینیمم مقدار خود را در $p = q$ بدست می‌آورند، که مقدار صفر برای واگرایی در KL و $H(p)$ است.
با توجه به توضیحات بالا آنتروپی دوتایی متقاطع برابر است با

$$L(\hat{y}, y) = [y.\log\hat{y} + (1-y).\log(1-\hat{y})]$$

که y در آن داده اصلی است و $\hat{y}$ داده ساخته شده است.
حال به نحوه‌ی محاسبه تابع هزینه [۷] برای شبکه مولد ناسازگار می‌پردازیم. در مرحله آموزش مدل تشخیص‌دهنده[۸]، برای برچسب داده‌های $x \in P_{data}(x)$ داریم $y = 1$ و $\hat{y} = D(x)$ پس داریم

$$L(D(x), 1) = \log(D(x)) \tag{۱.۳}$$

برای اینکه مدل تشخیص‌دهنده $\boldsymbol{D}$ بفهمد که ورودی از مدل تولیدکننده است، مقدار y را صفر قرار میدهیم. و خروجی مدل تشخیص دهنده $\boldsymbol{D}$ روی ورودی مدل تولیدکننده، بررسی می‌شود. پس برای داده‌های تولیدی مولد، به عنوان برچسب داده ها $y = 0$ و $\hat{y} = D(G(z))$ را همراه تولیدی قرار می‌دهیم.

$$L(D(G(Z))) = \log(D(x)) \tag{۲.۳}$$

---
relative entropy [۵]
training set [۶]
loss function [۷]
discriminator [۸]

برای اینکه مدل تشخیص‌دهنده داده های جعلی و اصلی را درست دسته‌بندی کند معادله‌های (۱.۳)و (۲.۳) باید ماکزیمم شوند. در این صورت تابع هزینه نهایی شبکه ناسازگار برابر است با

$$L^{(D)} = \max_D [\log D(x) + \log(1 - D(G(z)))] \tag{۳.۳}$$

برای بررسی تابع هزینه در شبکه مولد شرایط مشابه است، با این تفاوت که باید مدل تولید بگونه‌ای انتخاب شود که آنتروپی متقاطع مینیمم یا به عبارتی درست‌نمایی ماکزیمم گردد.

$$L^{(G)} = \min_G [\log D(x) + \log(1 - D(G(z)))] \tag{۴.۳}$$

با ترکیب معادلات (۳.۳) و (۴.۳) داریم

$$L = \min_G \max_D [\log D(x) + \log(1 - D(G(z)))] \tag{۵.۳}$$

معادله (۵.۳) تنها برای یک داده صحیح است برای اینکه آن را به کل مجموعه دادگان تعمیم دهیم،باید از L امید بگیریم. حال داریم

$$\min_G \max_D V(D,G) = \min_G \max_D (E_{x \sim P_{data}(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))]) \tag{۶.۳}$$

معادله (۶.۳) همان معادله‌ای است که "آین گود فلو" [۹] در مقاله شبکه‌های مولد ناسازگار خود در سال ۲۰۱۰ معرفی کرد.

```
# python's version of an Algorithm described in the original paper by Goodfellow et al. Minibatch
Stochastic gradient Descent traning of generative adverserial nets.

# k := The number of steps to apply to the discreminators {k=1 is least expensive}
for iteration in range(MAX_ITERATION):
    for step in range(k):
        # minibatch {z1, ... zm}, {x1, ..., xm}

        Z = [get_noise_from(generated_distribution) for _ in range(m)]
        X = [get_example_from(original_data_distribution) for _ in range(m)]
        update_discriminator(X, Z) # Ascending its stochastic gradient
    # minibatch {z1, ... zm}

    Z = [get_noise_from(generated_distribution) for _ in range(m)]
    update_generator(Z) # Descending its stochastic gradient
```

شکل ۱.۳: شبه کد آموزش شبکه مولد ناسازگار در زبان پایتون

[۹] *ian goodfellow*

از شبه کد موجود در تصویر ۱.۳ مشخص است که مدل مولد و مدل تشخیص‌دهنده به صورت مجزا آموزش داده می‌شوند. در قسمت اول داده‌های اصلی و جعلی با برچسب‌های مشخص به مدل تشخیص‌دهنده داده می‌شوند و فاز آموزش شروع می‌شود، حال گرادیان‌ها باز نشر داده می‌شوند در حالی که مدل مولد ثابت نگه داشته می‌شود. مدل تشخیص‌دهنده را با گرادیان تصادفی افزایشی[۱۰] بروز می‌کنیم، زیرا می‌خواهیم که مقدار تابع هزینه معادله (۶.۳) ماکزیمم شود. گرادیان افزایشی بدین‌گونه تعریف می‌گردد:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$$

از طرف دیگر، مدل مولد را در حالی که مدل تشخیص‌دهنده را ثابت نگه داشته‌ایم با تولید داده‌های جعلی و تنظیم کردن برچسب‌های تقلبی برای داده‌ها، به کمک گرادیان تصادفی کاهشی بروزرسانی می‌کنیم. این کار در جهت فریب دادن مدل تشخیص‌دهنده است. بهبود عملکرد مولد بر پایه پاداش یا جزایی است که مبتنی بر نتیجه مدل تشخیص دهنده است. علت اینکه از گرادیان تصادفی کاهشی[۱۱] استفاده کردیم، این بود که مقدار تابع هزینه مربوط به مدل مولد باید مینیمم شود. گرادیان کاهشی بدین‌گونه تعریف می‌گردد:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$$

۵.۳ شرایط بهینگی وتوقف

برای بررسی شرایط بهینگی از معادله (۶.۳) مشتق گرفته و آن را برابر صفر قرار میدهیم. در نتیجه به معادله زیر می‌رسیم.

$$-\frac{P_{data}(x)}{D(x)} + \frac{P_g(x)}{1 - D(x)} = 0 \Rightarrow D^*_G(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \tag{۷.۳}$$

حال که $\boldsymbol{D}$ بهینه را در دست داریم، آن را بدست آوردن هزینه $\boldsymbol{G}$ بهینه از معادله (۷.۳) جایگذاری می‌کنیم، بنابراین $L^{(G)}$ به صورت زیر خواهد بود

حالت متقارن و هموار شده‌ای از ***KL*** با تعریف زیر به نام ***Jensen-Shannon*** موجود است که می‌توانیم از آن به عنوان تعمیم ***KL*** استفاده کنیم.

$$JSD(P||Q) = \frac{1}{2}KL(P||\frac{P+Q}{2}) + \frac{1}{2}KL(Q||\frac{P+Q}{2})$$

حال با توجه به ***KL*** و حالت تعمیم پذیر شده آن (***Jensen-Shannon***) داریم

$$\Rightarrow KL(P_1||P_2) = Ex \sim P_1 \log \frac{P_1}{P_2}$$
$$\Rightarrow JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1+P_2}{2})$$
$$L^{(G)} = 2.JS(P_{data}||P_g) - 2\log 2$$

با نزدیک شدن P_{data} به P_g، واگرایی به سمت صفر می‌رود. یعنی P_{data} به P_g شبیه و شبیه‌تر می‌شود.

[۱۰] ***Stochastic Gradient Ascent***
[۱۱] ***Stochastic Gradient Descent***

$\lim_{P_{data} \to P_g} JS \to 0$ پس مینیمم مقدار $L^{(G)}$ برابر $-2\log 2$ خواهد بود.

از لحاظ ریاضی معادله (۷.۳) بسیار حائز اهمیت است ، اما در واقعیت نمی‌توانیم $\boldsymbol{D}$ بهینه را محاسبه کنیم چون توزیع آنرا را در دست نداریم. در واقع اگر توزیع آن را در دست داشیتم دیگر نیازی به این محاسبات نبود.

فصل ۴

چگونگی کارکرد یادگیری تقویتی عمیق

یادگیری تقویتی عمیق [۱] ، از شبکه‌های عصبی عمیق برای حل مسائل یادگیری تقویتی استفاده می‌کند، از این رو در نام آن از کلمه عمیق استفاده شده است. با در نظر گرفتن ***Q-Learning*** که یادگیری تقویتی کلاسیک محسوب می‌شود و ***Deep Q-Learning*** می‌توان تفاوت آن‌ها با یکدیگر را دید. در رویکرد اول، از الگوریتم‌های سنتی برای ساخت جدول Q استفاده می‌شود تا به عامل در یافتن اقدامی که باید در هر حالت انجام شود کمک کند. در دومین رویکرد، از شبکه عصبی (برای تخمین پاداش بر مبنای حالت: مقدار q) استفاده می‌شود.

۱.۴ نماد گذاری

در ادامه از نمادهای زیر برای انسجام متن استفاده خواهد شد.

فرایندهای تصمیم‌گیری مارکوف - یک فرایند تصمیم گیری مارکوف (به اختصار ***MDP***) شامل ۵ تایی $(S,A,\{P_{s,a}\},\gamma,R)$ است به طوری که:

$S :=$ ***set of States***

$A :=$ ***set of actions***

$\{P_{s,a}\} :=$ ***the state transition probabilities for*** $s \in S$ ***and*** $a \in A$

$\gamma \in [0,1] :=$ ***the discount factor***

$R : S \times A \longmapsto \Re$ ***or*** $R : S \longmapsto \Re :=$ ***reward function that the algorithm wants to maximize***

خط‌مشی - یک خط‌مشی π تابعی است $\pi : S \longmapsto A$ که حالات را به کنش‌ها نگاشت می‌کند. نکته: می‌گوییم ما در حال اجرای خط‌مشی π هستیم اگر به ازای وضعیت s کنش $a = \pi(s)$ را اجرا کنیم.

۲.۴ یادگیری تقویتی

یادگیری تقویتی براساس ایده فرضیه پاداش بنا شده است. به همین دلیل است که در یادگیری تقویتی برای داشتن بهترین رفتار، باید پاداش انباره‌ای بیشینه شود. پاداش انباره‌ای در هر گام t را می‌توان به صورت زیر نوشت:

$$G_t = \sum_{k=0}^{T} R_{t+k+1} \tag{۱.۴}$$

---

[۱] ***Deep Reinforcement Learning***

با این حال، در حقیقت نمی‌توان پاداش را به این شکل اضافه کرد. پاداشی که زودتر بیاید (در آغاز بازی) احتمال وقوع آن بیشتر است، زیرا از پاداش‌های بلند مدت آینده قابل پیش‌بینی‌تر هستند.

به همین خاطر یک نرخ تنزیل با عنوان گاما تعریف می‌شود. این مقدار باید بین ۰ و ۱ باشد.

هر چه گاما بزرگ‌تر شود، تنزیل کمتر است. این یعنی عامل یادگیرنده به پاداش‌های بلند مدت اهمیت بیشتری می‌دهد. از سوی دیگر، هرچه گاما کوچک‌تر باشد، تنزیل بیشتر است. این یعنی عامل توجه بیشتری به پاداش‌های کوتاه مدت می‌کند. پاداش مورد انتظار انباره‌ای تنزیل داده شده به صورت زیر محاسبه می‌شود:

$$G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \ \textbf{\textit{where}}\ \gamma \in [0,1) \tag{۲.۴}$$

۳.۴ وظایف اپیزودیک یا غیر اپیزودیک

وظایف اپیزودیک: در این نوع از وظایف، یک نقطه آغازین و یک نقطه پایانی (حالت ترمینال) وجود دارد. این منجر به ایجاد یک اپیزود یعنی لیست حالت‌ها، اعمال، پاداش‌ها و حالت‌های جدید می‌شود. برای مثال، در بازی برادران سوپرماریو، یک اپیزود با وارد شدن یک ماریو جدید آغاز می‌شود و هنگامی که ماریو کشته شد یا به پایان مرحله رسید، به پایان می‌رسد.

وظایف مستمر (غیر اپیزودیک): چنین وظایفی برای همیشه ادامه پیدا می‌کنند (فاقد حالت ترمینال). در این شرایط، عامل باید بیاموزد که چگونه بهترین عمل را انتخاب کرده و به طور همزمان با محیط نیز تعامل کند. وظیفه عاملی که به طور خودکار به تجارت سهام می‌پردازد، نمونه‌ای از وظایف مستمر است. برای این وظیفه، هیچ نقطه آغاز و حالت ترمینالی (نقطه پایان) وجود ندارد. عامل همواره به اجرا شدن ادامه می‌دهد تا کارشناس (کاربر انسانی) تصمیم به متوقف کردن آن بگیرد.

۴.۴ محاسبه تابع ارزش

برای سیاست π و وضعیت s ، تابع ارزش V^π را به صورت زیر تعریف می‌کنیم:

$$V^\pi(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots | s_0 = s, \pi] \tag{۳.۴}$$

۵.۴ الگوریتم تکرار ارزش

الگوریتم تکرار ارزش دو گام دارد:

۱) ارزش را مقداردهی اولیه می‌کنیم:

$$V_0(s) = 0 \tag{۴.۴}$$

۲) ارزش را با توجه به ارزش‌های قبلی تکرار می‌کنیم:

$$V_{i+1}(s) = R(s) + \max_{a \in A}[\sum_{s\prime \in S} \gamma P_{s,a}(s\prime) V_i(s\prime)] \tag{۵.۴}$$

۶.۴ روش یادگیری مونت کارلو و روش یادگیری *TD*

روش مونت کارلو [۲] : گردآوری پاداش‌ها در پایان هر اپیزود و محاسبه بیشینه پاداش آینده مورد انتظار

[۲] ***Monte Carlo Approach***

هنگامی که اپیزود به پایان می‌رسد (عامل به «حالت ترمینال» می‌رسد)، عامل به پاداش تجمعی کل نگاه می‌کند تا از نحوه عملکرد خود آگاه شود. در روش مونت کارلو، پاداش‌ها تنها در پایان بازی دریافت می‌شوند. بنابراین، بازی جدید با یک دانش افزوده آغاز می‌شود. عامل در هر تکرار تصمیمات بهتری اتخاذ می‌کند.

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{۶.۴}$$

با اجرای اپیزودهای بیشتر و بیشتر، عامل می‌آموزد که چگونه بهتر و بهتر عمل کند.

یادگیری تفاوت زمانی [۳] : تخمین پاداش در هر گام

در یادگیری ***TD*** ، عامل منتظر پایان اپیزود برای به روز رسانی تخمین بیشینه پاداش آینده نمی‌ماند. بلکه، تخمین مقدار ***V*** برای حالت‌های غیر ترمینال S_t به ایجاد شده در هنگام وقوع تجربه، به روز رسانی می‌شود. این روش

TD(0) یا ***TD*** یک گامی نام دارد (تابع ارزش پس از هر گام منفرد به روز رسانی می‌شود).

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{۷.۴}$$

روش‌های ***TD*** برای به روز رسانی مقدار تخمین، تنها منتظر گام بعدی می‌مانند. در زمان ***t+1*** آن‌ها بلافاصله یک هدف ***TD*** را با استفاده از پاداش مشاهده شده R_{t+1} و تخمین کنونی $V(S_{t+1})$ شکل می‌دهند. هدف ***TD*** یک تخمین است. در حقیقت عامل، تخمین پیشین $V(S_t)$ را با به روز رسانی آن طی گام‌های یک مرحله‌ای به روز رسانی می‌کند.

۷.۴ توازن جست‌وجو و استخراج (بهره‌برداری)

جست‌وجو [۴] ، یافتن اطلاعات بیشتر درباره محیط است.

استخراج [۵] ، بهره‌برداری از اطلاعات شناخته شده برای بیشینه‌سازی پاداش است.

به خاطر داشته باشید که هدف عامل ***RL*** بیشینه‌سازی پاداش انباره‌ای است. با این حال می‌توان در یک تله متداول افتاد. . در یک بازی، عامل می‌تواند میزان نامتناهی تشویق‌های کوچک داشته باشد (۱+ برای هر کدام). اما در جای دیگر زمین بازی، حجم زیادی تشویق وجود دارد (۱۰۰۰+). اگر فقط روی پاداش تمرکز شود، عامل هیچ وقت به آن کوه عظیم پاداش نمی‌رسد. در عوض، تنها نزدیک‌ترین منابع پاداش را جست‌وجو می‌کند، حتی اگر این منابع کوچک باشند. اگر عامل، مقدار کمی جست‌وجو کند، می‌تواند پاداش بزرگی پیدا کند. این همان چیزی است که از آن با عنوان «توازن جست‌وجو و استخراج» نام برده شده است. کارشناس باید قوانینی را تعیین کند تا به برقراری توازن و مدیریت آن کمک کند.

۸.۴ سه رویکرد به یادگیری تقویتی

رویکرد ارزش محور: در رویکرد ارزش محور، هدف بهینه‌سازی تابع ارزش ***V(s)*** است. تابع ارزش، تابعی است که پاداش بیشینه آینده را مشخص می‌کند که عامل در هر حالت دریافت می‌کند. ارزش هر حالت برابر است با ارزش

Temporal Difference Learning or TD [۳]
Exploration [۴]
Exploitation [۵]

کل پاداشی که عامل می‌تواند انتظار داشته باشد در آینده با آغاز از آن حالت جمع‌آوری کند. عامل از این تابع ارزش برای انتخاب آنکه کدام حالت در هر گام انتخاب شود، استفاده می‌کند. عامل حالتی با بیشترین ارزش را انتخاب می‌کند.

$$V_\pi = E_\pi[\sum_{k=0} \gamma^k R_{t+k+1} | S_t = s] \tag{۸.۴}$$

رویکرد سیاست محور: در یادگیری تقویتی سیاست‌محور، قصد بهینه‌سازی تابع سیاست $\pi(s)$ بدون استفاده از تابع ارزش است. سیاست چیزی است که رفتار عامل را در یک زمان داده شده، تعیین می‌کند. عامل یک تابع سیاست را می‌آموزد. این امر به او کمک می‌کند تا هر حالت را به بهترین عمل ممکن نگاشت کند. دو دسته از سیاست‌ها وجود دارند:

قطعی: سیاست برای یک حالت داده شده همیشه عمل مشابهی را باز می‌گرداند.

تصادفی: برای هر یک از اعمال یک توزیع احتمالی در نظر می‌گیرد.

$$\pi(a|s) = P[A_t = a | S_t = s] \tag{۹.۴}$$

رویکرد مدل محور: در یادگیری تقویتی مدل محور، محیط مدل می‌شود. این یعنی مدلی از رفتار محیط ساخته می‌شود. مساله مهم در این رویکرد نیاز به مدل متفاوت برای ارائه هر محیط است.

۹.۴ Q - یادگیری

کیو-یادگیری تکنیک یادگیری تقویتی است که با یادگیری یک تابع اقدام/مقدار، سیاست مشخصی را برای انجام حرکات مختلف در وضعیت‌های مختلف دنبال می‌کند. یکی ازنقاط قوت این روش، توانایی یادگیری تابع مذکور بدون داشتن مدل معینی ازمحیط می‌باشد . اخیراً در این روش اصلاحی با نام کیو-یادگیری تاخیری انجام شده که بهبود قابل توجهی ایجاد نموده است. در روش اخیر یادگیری ***PAC*** با فرایندهای تصمیم مارکوف ترکیب شده‌اند.

در اینجا مدل مسئله تشکیل شده از یک عامل، وضعیت‌ها S و مجموعه از اقدامات A برای هر وضعیت.با انجام یک اقدام $a \in A$ ، عامل از یک وضعیت به وضعیت بعدی حرکت کرده و هر وضعیت پاداشی به عامل می‌دهد. هدف عامل حداکثر کردن پاداش دریافتی کل خود است. این کار با یادگیری اقدام بهینه برای هر وضعیت انجام می گردد. الگوریتم دارای تابعی است که ترکیب حالت/اقدام را محاسبه می نماید :

$$Q : S \times A \to \Re \tag{۱۰.۴}$$

قبل از شروع یادگیری، Q مقدار ثابتی را که توسط طراح انتخاب شده برمی گرداند. سپس هر بار که به عامل پاداش داده می‌شود، مقادیر جدیدی برای هر ترکیب وضعیت/اقدام محاسبه می گردد. هسته الگوریتم از یک بروز رسانی تکراری ساده تشکیل شده‌است. به این ترتیب که بر اساس اطلاعات جدید مقادیر قبلی اصلاح می‌شود.

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R(s_t)}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{max future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right]$$

که $R(s_t)$ پاداش s_t و $\alpha_t(s,a)$ است. نرخ یادگیری ($0 \leq \alpha \leq 1$) ممکن است برای همه زوج‌ها یکسان باشد. مقدار عامل تخفیف γ بگونه است که $0 \leq \gamma \leq 1$

فرمول فوق معادل عبارت زیر است:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t)(1 - \alpha_t(s_t, a_t)) + \alpha_t(s_t, a_t)[R(s_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})] \quad (۱۱.۴)$$

یک اپیزود الگوریتم وقتی s_{t+1} به وضعیت نهایی می سد پایان می یابد. توجه کنید که برای همه وضعیت‌های نهایی s_f و $Q(s_f, a)$ مربوطه هیچگاه بروز نمی‌شود و مقدار اولیه خود را حفظ می‌کند.

فصل ۵

بررسی عملکرد مدل ارائه شده

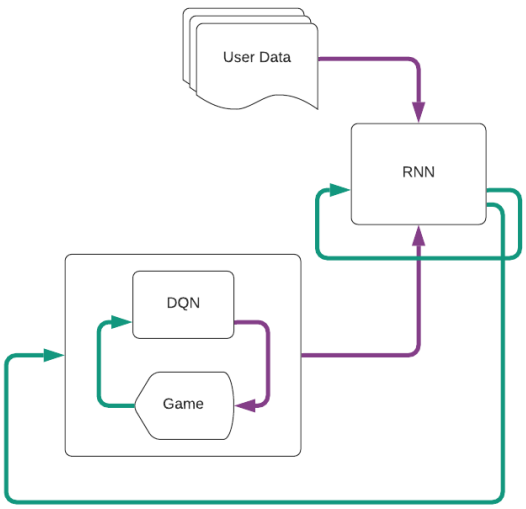
۱.۵ ساختار مدل

همانطور که گفته شد این مدل از چارچوب شبکه‌های عصبی مولد ناسازگار استفاده می‌کند. شبکه تشخیص دهنده از یک شبکه عصبی پیشخور ساخته شده و شبکه فریب دهنده از یک شبکه عصبی یادگیری عمیق ساخته شده است.

شبکه یادگیری عمیق به طور مستقیم با بازی در ارتباط است و با بازی کردن و بازخورد گرفتن از بازی آموزش میبینند. پس از آموزش دیدن و بازخورد گرفتن از بازی، خروجی حاصل از بازی کردن شبکه، با کد گذاری مخصوص استخراج شده و در اختیار شبکه تشخیص دهنده گذاشته می‌شود.

شبکه تشخیص دهنده خروجی حاصل از بازی کردن کاربران را در کنار شبکه فریب دهنده می‌بیند و باید در نهایت تشخیص دهد که هر یک واقعی هستند یا خیر.

در نهایت بعد از مشخص شدن خروجی شبکه تشخیص دهنده، بازخوردی به شبکه فریب دهنده و خود شبکه تشخیص دهنده داده می‌شود.این بازخورد برای تقویت شبکه مورد استفاده قرار میگیرد. تصویر ۱.۵ نشان دهنده شمای کلی این مدل است.



شکل ۱.۵: نمودار کلی مدل ارائه شده

۲.۵ نحوه کدگذاری یک تجربه بازی

بازی انتخاب شده به بازی ***pong*** معروف است که بازی بسیار ساده‌ای است. این بازی برای کاهش پیچیدگی بخش تبدیل و کدگذاری انتخاب شده است، اما این مدل به شرط یافتن کدگذاری مناسب، می‌تواند برای هر بازی‌ای کارا باشد.

کدگذاری به این صورت اتفاق افتاده است که در هر ثانیه، ۵ بار دکمه فشرده شده توسط کاربر را ذخیره‌سازی میکند. تعداد بیشتر از ۵، اندازه داده را بزرگتر می‌کند اما لزوما اطلاعات بیشتر نمی‌شود. سرعت بازخورد انسان به یک اتفاق و تغییر نیز در حدود ۲۰۰ میلی ثانیه است، پس این دقت تا حدود خوبی تمام رفتار کاربر را ثبت می‌کند.

بازی ***pong*** از دو دکمه جهت ساخته شده است که سکو را به چپ و راست هدایت می‌کند. پس در هر بار که داده گرفته می‌شود، یکی از حالات چپ، راست یا بی‌حرکت خواهد بود که با حروف ***L, R, S*** نمایش داده می‌شوند که در آن ***S*** نماینده ثابت بودن است.

برای هر گل که بازیکن می‌زند نماد ***G*** و گل خورده ***D*** خواهد بود. بعد از هر گل بازی نیز دوباره از نو شروع می‌شود. به موازات ثبت حرکات بازی‌کن، مختصات توپ نیز ثبت می‌شود. برای آموزش مدل، این مختصات ضروری‌است. توجه شود که در بازی، سرعت حرکت توپ و سکوها برابر نیست و طوری تنظیم شده‌است که اگر بازیکن سریع عمل نکند، توپ را از دست می‌دهد.

هر بازی در نهایت یک رشته حرکات و یک آرایه از مختصات توپ خواهد بود.

۳.۵ بررسی روند فاز آموزش

هر مرحله از آموزش شامل یک حلقه یادگیری داخلی برای شبکه فریب‌دهنده، یک خروجی از تجربه بازی و یک حلقه یادگیری شبکه تشخیص دهنده است.

شبکه فریب دهنده داخل بازی قرار میگیرد و در برابر خودش. هر بازی برای این شبکه دو تجربه بازی محسوب می‌شود و خود را مدام بهبود می‌بخشد. در این مرحله شبکه قوانین بازی را می‌آموزد و اینکه چقدر شبیه به انسان است اهمیتی ندارد، تنها هدف پیروزی است.

پس از پشت سر گذاشتن مرحله یادگیری شبکه فریب دهنده، یک بار دیگر شبکه را داخل بازی قرار می‌دهیم و تمام حرکات را با شیوه کدگذاری ذکر شده ثبت میکنیم و به عنوان ورودی در کنار داده‌های جمع آوری شده از بازیکنان واقعی گذاشته و مرحله یادگیری شبکه تشخیص دهنده را شروع میکنیم.

تشخیص دهنده یک تجربه یک دقیقه‌ای از بازی را به عنوان ورودی می‌گیرد و در خروجی دو حالت واقعی یا فریب را نسبت می‌دهد. ورودی شبکه به دو دلیل محدود به یک دقیقه شده است، اول اینکه ورودی شبکه می‌بایستی مشخص باشد که چند گره دارد و دوم آنکه یک تشخیص دهنده باید در یک زمان مناسب فریب دهنده را تشخیص دهد تا در دنیای واقعی نیز، زمانی که در بازی قرار میگیرد، حتی قبل از پایان بازی فریب دهنده را تشخیص دهد.

پس از آنکه کار تشخیص دهنده تمام شد، یک بازخورد به شبکه فریب دهنده داده می‌شود. تا بتواند خود را به رفتار یک انسان نزدیک کند.

وجود حلقه اول، از آن جهت است که اگر تنها شبکه فریب دهنده را با داده‌های ورودی مقایسه کنیم و شبکه از خود بازی بازخورد نگیرد، ممکن است رفتارش بسیار شبیه انسان بشود، اما بر اساس قوانین بازی برخورد نکند! البته این مسئله در این بازی که ورودی‌ها و قوانین ساده‌ای دارد اتفاق نمی‌افتد.

۴.۵ برخی نواقص و برتری‌های مدل

یکی از اصلی‌ترین مشکل‌هایی که این مدل دارد این است که با توجه به داده‌های انسانی، میتواند ربات‌هایی در سطوحی متفاوت از مبتدی تا حرفه‌ای بسازد و نمی‌توان از نوع خروجی مطمئن بود. مگر آنکه داده انسانی برچسب خورده باشد که این کار هزینه و زمان نیاز دارد.

از طرفی پیدا کردن یک شیوه کدگذاری برای تفهیم به شبکه تشخیص دهنده، کاری دشوار است و ممکن است برای هر بازی‌ای این امر ساده و یا حتی ممکن نباشد.

بزرگترین برتری این مدل آن است که خروجی شباهت بسیار زیادی به انسان دارد و استفاده آن در بازی تجربه‌ای بسیار طبیعی را میسازد.

فصل ۶

نتیجه گیری و پیشنهادات

۱.۶ یافته های تحقیق

در بخش‌های ابتدایی این پروژه با مفاهیم کلی پیرامون شبکه‌های عصبی و بخش‌بندی‌های آنها آشنا شدیم،سپس به صورت تخصصی‌تر به سراغ شبکه‌های مولد ناسازگار رفته و کمی با دانش ریاضیات و آمار استفاده شده در آنها آشنا شدیم. همچنین مفاهیم پایه‌ای یادگیری تقویتی را مطالعه و بررسی کردیم.

در این پروه سعی شد مفاهیم مربوط به شبکه‌های عصبی مولد ناسازگار در بحث تولید بات‌های بازی انسان‌نما بررسی شوند. در نهایت با بیشتر کردن داده‌ها و افزایش تعداد حلقه‌ها می‌توان این مدل را بهبود داد و با تکرارهای بیشتر به دقت بیشتری رسید. متاسفانه برای تشخیص عملکرد شبکه تشخیص دهنده محیط مناسبی در دسترس نبود و عملکرد و بازدهی آن قابل مطالعه و بررسی نبود.

۲.۶ ارائه پیشنهادات

برای بررسی دقیق عملکرد شبکه فریب دهنده، میتوان به موازات آن یک بات دیگر نیز با روش‌های دیگر آموزش داد که شیوه بازی کردن آنان و عملکردشان با هم مقایسه شود.

همچنین استفاده از انواع بات میتواند برای بررسی عملکرد شبکه تشخیص دهنده مورد استفاده قرار گیرد.

با برچسب گذاری داده بازیکنان واقعی حتی میتوان بات‌هایی با درجه سختی متفاوت ساخت و از آن‌ها استفاده کرد.

این مدل ارائه شده یک مدل علمی است و برای کاربردی شدن باید درون یک بازی واقعی پیاده‌سازی شده و چالش‌های دنیای واقعی نیز بررسی شوند.

فهرست مراجع

[1] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., & Bengio, Y. (2014). Generative Adversarial Nets. NIPS.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. (2013). Playing Atari with Deep Reinforcement Learning. NIPS.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence. Research, 47:253–279, 2013.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1106–1114, 2012.

[5] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In Neural Networks (IJCNN), The 2010 International Joint Conference on, pages 1–8. IEEE, 2010.

[6] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

[7] Andrew Moore and Chris Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. Machine Learning, 13:103–130, 1993.

[8] Richard Sutton and Andrew Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

[9] Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58–68, 1995.

[10] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8(3-4):279–292, 1992.

Abstract

Human-like bots are some kind of robots that can do human tasks, perform like them and it's intended to be designed with minimum difference in comparison to humans. In digital world, confronting with these bots are in two ways: sometimes we want to recognise destructive bots in a product, sometimes we want to use bots to replace human resource to lower the cost and the more bots look like human, the more customer satisfaction we gain.

In game industry, both forms are used. First form used to recognise cheating players, and the second form is to develop an AI for player training.

The method we used in this project, profits from GAN to solve both of these problems.

Keywords:

Human-like bot, Neural Network, Generative Adversarial Network, Reinforcement Learning, Bot Detection

Tehran University

Department of Math, Statistics and Computer Science

BSc Thesis

Artificial IntelligenceField

Title

Using GAN Framework for Making Human-like Game Bots

By

Parsa Yahooie

Supervisor

Dr. Hedieh Sajedi

February 2021