هفتمین کنفرانس بین‌المللی
«بازی‌های رایانه‌ای؛ فرصت‌ها و چالش‌ها»
۵ و ۶ اسفند ۱۴۰۰ – دانشگاه اصفهان

بهبود تولید مراحل بازی Super Mario Bros با استفاده از الگوریتم‌های نمونه‌برداری در زبان برنامه‌سازی احتمالاتی فیگارو

زهرا امیرجان[۱،*] و مرتضی دُرّی‌گیو[۲]

۱- **کارشناسی ارشد هوش مصنوعی، دانشگاه سمنان**
zahra.amirjan@semnan.ac.ir

۲- **استادیار دانشکده مهندسی برق و کامپیوتر، دانشگاه سمنان**
dorrigiv@semnan.ac.ir

چکیده

از وظایف اصلی در توسعه یک بازی ویدئویی، تولید محتوا از جمله مراحل بازی، داستان‌ها، اجزای گرافیکی و غیره می‌باشد. چنین فرآیند تولیدی، به دلیل پیچیدگی و نیاز به نیروی انسانی متخصص، هزینه‌ی زیادی را به توسعه بازی ویدئویی تحمیل می‌نماید. از این‌رو، تولید محتوای رویه‌ای، به عنوان روشی مناسب برای کاهش هزینه‌ها با استفاده از تکنیک‌های الگوریتمی برای تولید خودکار برخی از محتویات بازی از جمله مراحل بازی، مورد توجه بوده است. مدل‌های مارکوف به عنوان یک روش تولید محتوای رویه‌ای مبتنی بر احتمالات، برای مدل‌سازی و تولید محتوای بازی‌ها استفاده می‌شود. توسعه یک مدل احتمالاتی به‌طور معمول، نیازمند ایجاد نمایش برای مدل و الگوریتم استدلال است که می‌تواند نتایج مفید را از شواهد دریافت کند و در بسیاری موارد، الگوریتمی برای یادگیری جنبه‌های مدل از داده‌ها فراهم آورد. زبان‌های برنامه‌نویسی احتمالاتی، با ترکیب احتمالات به‌همراه قدرت محاسباتی زبان‌های برنامه‌نویسی می‌توانند برای ساخت مدل‌های احتمالاتی و استدلال روی آن‌ها مفید واقع شوند. زیرا ساخت مدل‌های احتمالاتی به‌صورت مستقیم، کاری سنگین است و پیاده‌سازی آن غالباً مشکلات زیادی به‌همراه دارد. در این مقاله، از یک زبان برنامه‌نویسی احتمالاتی به نام Figaro برای مدل‌سازی زنجیره‌های چندبعدی مارکوف برای تولید خودکار مراحل بازی Super Mario Bros استفاده شده است. هدف اصلی روش پیشنهادی، امکان استنتاج بهتر و کاراتر با استفاده از الگوریتم‌های استنتاج متنوع موجود در Figaro، بر روی مدل احتمالاتی برای تولید مراحل بازی می‌باشد.

کلمات کلیدی: تولید محتوای رویه‌ای، مدل‌های مارکوف، برنامه‌نویسی احتمالاتی، Figaro

۱–مقدمه

هزینه تولید محتوای بازی‌های ویدئویی بخش عمده‌ای از هزینه یک پروژه تولید بازی را شامل می‌شود و هم‌چنین دشواری‌های زیادی را به‌همراه دارد. معمولاً یک تیم متشکل از بخش‌های تولیدی مختلف بازی (برنامه‌نویسان، مهندسان صدا، هنرمندان، و غیره)، مسئول ایجاد همه محتوای بازی هستند. این موضوع تأثیر قابل توجهی بر میزان بودجه اختصاص یافته برای توسعه، به خصوص زمانی که ضرورت ایجاد یک بازی با کیفیت بالا وجود داشته باشد؛ خواهد داشت. از این‌رو تولید محتوای رویه‌ای

(PCG[1]) به عنوان یک تکنیک مناسب برای بهینه‌سازی فرآیند تولید و کاهش هزینه‌ها با استفاده از تکنیک‌های الگوریتمی برای تولید خودکار مراحل یک بازی ویدئویی مورد استفاده قرار گرفته است. از سوی دیگر، تولید محتوای رویه‌ای یک منطقه روبه‌رشد تحقیقاتی است که بر روی استفاده از هوش مصنوعی در طراحی و ایجاد محتوا و اغلب برای بازی‌های ویدئویی متمرکز شده است. زنجیره‌های چندبعدی مارکوف (MdMCs[2])، یکی از روش‌های تولید مراحل بازی مبتنی بر یادگیری ماشین است که از احتمالات برای یادگیری استفاده می‌کند [۱]. در این روش، برای ضبط احتمال انتقال کاشی[3] به کاشی از مراحل آموزش بهره گرفته و سپس از آن احتمالات آموزش یافته برای نمونه‌برداری تولید مرحله جدید استفاده می‌کند. در این مقاله، تکنیک زنجیره‌های چندبعدی مارکوف برای تولید مراحل بازی دوبعدی و پلتفرم Super Mario Bros مورد بحث قرار گرفته است. البته پژوهش‌های دیگری نیز در زمینه تولید مراحل بازی Super Mario Bros وجود دارند که خارج از محدوده بحث این مقاله هستند [۲–۶].

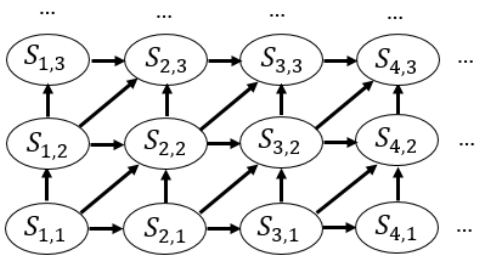
مدل‌های احتمالی، متغیرهای تصادفی و توزیع‌های احتمال را در مدل یک رویداد یا پدیده قرار می‌دهند. درحالی‌که یک مدل قطعی، یک نتیجه ممکن واحد برای یک رویداد را نشان می‌دهد، خروجی یک برنامه احتمالاتی یک توزیع احتمال است که به برنامه‌نویس اجازه می‌دهد تا به‌طور مشخص عدم قطعیت مربوط به یک نتیجه را تجسم کند. زبان‌های برنامه‌نویسی احتمالاتی (PPLs[4])، برای توصیف مدل‌های احتمالی، طراحی شده است و سپس در این مدل‌ها با استفاده از الگوریتم‌های استنتاج، نتیجه‌گیری و استنتاج می‌کنند. در نتیجه این زبان‌ها می‌توانند با دخالت بسیار کم توسعه‌دهنده نتیجه‌ی مورد نظر را بازگردانند. این زبان‌های برنامه‌نویسی به‌طور ویژه به مدل‌های گرافی (شبکه‌های بیزین و شبکه‌های مارکوف) وابسته هستند، اما در مقابل از انعطاف‌پذیری بیشتری برخوردارند. زبان‌های برنامه‌نویسی احتمالاتی اغلب از یک زبان پایه گسترش می‌یابند. انتخاب زبان پایه بستگی به شباهت مدل به آنتولوژی (هستی‌شناسی) آن زبان، ملاحظات تجاری و هم‌چنین اولویت‌های خاص دامنه هدف دارد. به‌عنوان مثال، زبان برنامه‌نویسی فیگارو[5] از اسکالا[6] گسترش می‌یابد [۷]. فیگارو از تعدادی الگوریتم‌های استدلال داخلی ساخته شده است که می‌توانند به‌طور خودکار به مدل‌های جدید اعمال شوند. علاوه بر این، مدل‌های فیگارو، ساختار داده‌ها در زبان برنامه‌نویسی اسکالا هستند که با جاوا سازگار هستند و می‌توان آن‌ها را ساخت، مورد تغییر قرار داد و مستقیماً در هر برنامه اسکالا یا جاوا استفاده کرد. در یادگیری ماشینی سنتی، برای افزایش کارایی، اقدام به جمع‌آوری داده‌های بیشتر و بیشتر می‌شود و به ماشین اجازه داده می‌شود تا یاد بگیرد و کار کند. در زبان‌های برنامه‌نویسی احتمالاتی اصول و زیربنای سیستم‌ها بر دانش بیشتر استوار است. در نتیجه، برنامه‌نویسی احتمالاتی می‌تواند راه را برای ساده کردن یادگیری ماشینی فراهم کند.

۲–بیان مسأله

مدل‌های مارکوف روابط احتمالاتی بین متغیرها را در بر می‌گیرند. یک زنجیره مارکوف به عنوان یک مجموعه‌ای از حالت[7]ها به صورت $S = \{s_1, s_2, \dots, s_n\}$ و توزیع احتمال شرطی (CPD[8]) به صورت $p(s_t|s_{t-1})$ که نشان‌دهنده‌ی احتمال انتقال

[1] Procedural Content Generation
[2] Multi-dimensional Markov Chains
[3] Tile
[4] Probabilistic Programming Languages
[5] Figaro
[6] Scala
[7] State
[8] Conditional Probability Distribution

به حالت $s_t \in S$ از حالت قبلی آن یعنی $s_{t-1} \in S$ می‌باشد، تعریف می‌شود. زنجیره‌ی مارکوف استاندارد به این صورت است که احتمال هر حالت، فقط به یک حالت قبلی آن وابسته بوده است. ولی زنجیره‌ی مارکوف مرتبه بالاتر هم وجود دارد، مانند زنجیره مارکوف مرتبه d، که در آن هر حالت به d حالت قبلی آن وابسته می‌باشد (d یک عدد طبیعی می‌باشد). در این حالت، توزیع احتمال شرطی آن به صورت $p(s_t|s_{t-1}, \dots, s_{t-d})$ می‌باشد. زنجیره‌های مارکوف چندبعدی یک تعمیم از زنجیره‌های مارکوف مرتبه بالا است که ساختار شبکه آن به صورت یک گراف چندبعدی از حالت‌ها می‌باشد. به عنوان مثال، توزیع احتمال شرطی زنجیره مارکوف چندبعدی موجود در شکل ۱، می‌تواند به صورت $p(s_{t,r}|s_{t-1,r}, s_{t,r-1}, s_{t-1,r-1})$ نوشته شود (t و r نشان‌دهنده‌ی ابعاد شبکه می‌باشند که t بیانگر شماره ستون و r بیانگر شماره ردیف می‌باشد).



شکل ۱: ساختار شبکه زنجیره مارکوف چندبعدی مرتبه ۳ (بازتولید شده از [۱])

شکل ۱، یک ساختار شبکه برای زنجیره‌ی مارکوف چندبعدی مرتبه ۳ می‌باشد. استفاده از زنجیره چندبعدی به جای یک‌بعدی، مدل را قادر می‌سازد تا به راحتی روابط را از داده‌های آموزشی دوبعدی از قبیل بافت‌های گرافیکی یا مراحل بازی‌های ویدئویی دریافت کند.

۲–۱– آموزش مدل MdMC و نمونه‌برداری از مدل آموزش دیده

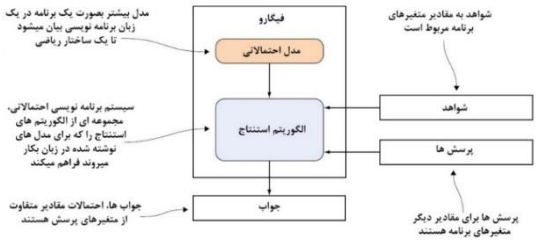
در پژوهش پیشین [۸]، نحوه نمایش داده‌های آموزشی (یعنی نقشه‌های بازی) و هم‌چنین الگوریتم آموزش مدل MdMC در تولید مراحل بازی Super Mario Bros و نحوه‌ی آموزش مدل که چگونه بایستی یک جدول احتمال برای یک ساختار شبکه خاص برای مدل MdMC را از فرکانس رخدادها در مراحل آموزش ایجاد نمود، به‌طور کامل شرح داده شده است. در مدل MdMC، نمونه‌برداری از یک مرحله جدید به معنی پرکردن یک ماتریس با کاشی‌های یک موقعیت در یک زمان با نمونه‌برداری از مدل آموزش دیده می‌باشد. گاهی اوقات نمونه‌برداری یک مرحله جدید به این شیوه، منجر به پیکربندی کاشی‌ها خواهد شد که داده‌ای برای آن وجود ندارد. در نتیجه، به این وضعیت یک حالت نادیده[9] گفته می‌شود. حالت‌های نادیده، نامطلوب هستند، زیرا هیچ داده‌ی آموزشی برای آن‌ها وجود نداشته و در نتیجه در مواجهه با این حالت‌ها مجبور به انتخاب یک کاشی به صورت تصادفی خواهیم بود. برای جلوگیری از این مسئله، الگوریتم از روش look-ahead (که از این پس با l نشان داده می‌شود) استفاده می‌کند که به نمونه‌برداری چندین کاشی جلوتر منجر می‌شود تا جایی که تضمین کند که به هیچ حالت نادیده نرسد. اگر یک حالت نادیده با ساختار شبکه فعلی اجتناب ناپذیر باشد، الگوریتم به ساختار ساده‌تر برمی‌گردد. وقتی از یک ساختار شبکه ساده‌تر استفاده می‌شود، احتمال رخ دادن یک حالت نادیده کاهش می‌یابد. دلیل این موضوع این است که در وضعیت جدید، هر حالت به تعداد حالت قبلی کمتری وابسته هست. الگوریتم نمونه‌برداری مدل MdMC و نحوه‌ی نمونه‌برداری یک مرحله جدید از مدل آموزش دیده به‌طور کامل در [۸] شرح داده شده است.

[9] Unseen state

۳–برنامه‌نویسی احتمالاتی

یک برنامه‌ی احتمالاتی، ترکیبی از محاسبات قطعی به همراه ورودی‌های تصادفی است که این محاسبات باعث ایجاد یک شمای کلی درمورد داده‌ها می‌گردند. احتمالات به صورت ضمنی در این نتایج دخیل شده‌اند و نیازی به پیاده‌سازی فرمول‌های مختلف نیست. زبان‌های برنامه‌نویسی احتمالاتی، یکسری الگوریتم استنتاج عمومی ارائه می‌دهند که می‌توانند با دخالت بسیار کم توسعه‌دهنده نتیجه‌ی مورد نظر را بازگردانند. به عبارت دیگر، این ویژگی اجازه می‌دهد تا تیم‌های مدل‌سازی و متخصصین بررسی داده‌ها را از هم مجزا نمود. یک رویکرد خوب برای استدلال تحت شرایط عدم قطعیت، استدلال احتمالاتی است. فیگارو برای کمک به ساختن و استنتاج طیف گسترده‌ای از مدل‌های احتمالی طراحی شده است. به عنوان مثال می‌توان به مدل‌های جهت‌دار و بدون جهت، مدل‌هایی که در آن، شرایط و محدودیت‌ها توسط توابع اسکالا دلخواه بیان می‌شود، مدل‌هایی که شامل اشیاء مرتبط با یکدیگر هستند، مدل‌های جهان باز (open universe) که در آن ما نمی‌دانیم چه چیزی و چه مقدار اشیا وجود دارد، مدل‌های شامل عناصر گسسته و پیوسته، مدل‌هایی که در آن عناصر ساختارهای داده‌ای غنی مانند درخت هستند، مدل‌هایی با تصمیمات ساختاری (structured decisions) و مدل‌های با پارامترهای ناشناخته و نامعلوم اشاره کرد [۹]. استدلال احتمالاتی نیازمند تلاش و تخصص قابل توجه است. علاوه براین، بیشتر ابزار استدلال احتمالاتی برای ادغام به برنامه‌های بزرگتر، مستقل و دشوار است. فیگارو، یک زبان برنامه‌نویسی احتمالاتی است که به هر دو این مسائل کمک می‌کند. فیگارو امکان بیان مدل‌های احتمالی را با استفاده از قدرت زبان‌های برنامه‌نویسی فراهم می‌کند و به مدل‌ساز، ابزار بیان برای ایجاد انواع مدل‌ها را می‌دهد.

شکل ۲ رابطه بین سیستم‌های برنامه‌نویسی احتمالاتی و سیستم‌های استدلال احتمالاتی را به‌طور کلی نشان می‌دهد. همان‌طور که ملاحظه می‌شود، مدل‌ها بیشتر از اینکه یک ساختار ریاضی شبیه به یک شبکه بیزی باشند، به عنوان برنامه‌هایی در یک زبان برنامه‌نویسی بیان می‌شوند. در نتیجه این تغییر، شواهد، نمایش‌ها و پرسش‌ها همه به متغیرهای برنامه اعمال می‌شوند. شواهد ممکن است مقادیر خاصی را برای متغیرهای برنامه مشخص کنند، پرسش‌ها برای مقادیر متغیرهای برنامه پرسش می‌شوند، و جواب‌ها احتمالات مقادیر متفاوتی از متغیرهای پرسش می‌باشند. به علاوه، یک سیستم برنامه‌نویسی احتمالاتی معمولاً با مجموعه‌ای از الگوریتم‌های استنتاج همراه است. این الگوریتم‌ها برای برنامه‌های نوشته شده در زبان بکار می‌روند.



شکل ۲: معماری مربوط به سیستم برنامه‌نویسی احتمالاتی (بازتولید شده از [۱۰])

عناصر، تمام ساختارهای داده‌ای یک مدل فیگارو هستند. عنصر، یک ساختار داده است که نمایانگر فرآیندی است که به‌طور تصادفی یک مقدار تولید می‌کند. یک فرآیند تصادفی می‌تواند به هر تعداد نتیجه منجر شود که هر نتیجه ممکن به عنوان ارزش فرآیند شناخته می‌شود. برای تولید عناصر پیچیده‌تر می‌توان عناصر را به روش‌های مختلفی ترکیب کرد. عناصر فیگارو که در آزمایشات این مقاله به کار گرفته شده است و ساختار آن‌ها و هم‌چنین کاربرد استفاده از آن‌ها در مرجع [۱۰] به‌طور کامل شرح داده شده است.

۴–روش پیشنهادی

الگوریتم نمونه‌برداری یک مرحله جدید با استفاده از برنامه‌نویسی فیگارو مشابه با الگوریتم نمونه‌برداری روش مارکوف چندبعدی عمل می‌کند، با این تفاوت که در برنامه‌نویسی فیگارو به جای متغیرها از عناصر استفاده می‌شود و عناصر، ساختار مدل احتمالاتی را می‌سازند. علاوه بر این موضوع، در نمونه‌گیری یک مرحله جدید با استفاده از برنامه‌نویسی فیگارو از الگوریتم‌های استنتاج فیگارو استفاده می‌شود تا محتمل‌ترین حالت برای هر موقعیت در نقشه بازی تولید شود. الگوریتم نمونه‌گیری یک مرحله جدید از بازی Super Mario Bros با استفاده از برنامه‌نویسی فیگارو در پژوهش قبلی [۸] قرار گرفته است و طرز کار این الگوریتم برای نمونه‌برداری یک مرحله جدید از بازی به‌طور کامل شرح داده شده است. لازم به ذکر است که در این الگوریتم از الگوریتم استنتاج MPEVariableElimination استفاده شده است که در این مقاله علاوه بر آن، الگوریتم‌های MPEBeliefPropagation و MetropolisHastingsAnnealer نیز مورد استفاده قرار گرفته‌اند.

۴–۱– استنتاج روی مدل احتمالاتی در برنامه‌نویسی با Figaro

هنگامی که یک برنامه احتمالی نوشته می‌شود و یک مدل احتمالی توسط آن ایجاد می‌شود، می‌توان انواع الگوریتم‌های استنتاج را روی آن مدل‌ها اعمال کرد. فیگارو شامل تعدادی الگوریتم استدلال است که این امکان را فراهم می‌کند تا با مدل‌های احتمالی کارهای مفیدی انجام شود. دو نوع الگوریتم استدلال وجود دارد: ۱–Factored algorithms: این الگوریتم‌ها بر روی ساختارهای داده به نام فاکتورها کار می‌کند که روی مدل احتمالاتی استدلال انجام می‌دهد. یک فاکتور نمایشی از یک تابع از مقادیر مجموعه‌ای از متغیرها به اعداد واقعی می‌باشد. ۲–Sampling algorithms: این دسته الگوریتم‌ها نمونه‌هایی از جهان‌های ممکن را بر اساس توزیع‌های احتمال تولید کرده و از این نمونه‌ها استفاده می‌کند تا پاسخ پرس‌وجو (query) را درباره‌ی مدل احتمالاتی بدهد. بسیاری از برنامه‌ها متغیرهای زیادی دارند و این متغیرها می‌توانند دارای انواع داده‌ای غنی با تعداد زیادی از مقادیر باشند. در آن برنامه‌ها، حتی ایجاد همه عوامل لازم برای اجرای یک الگوریتم استنتاج فاکتور ممکن نیست. با استفاده از نمونه‌گیری، فقط باید تعداد نسبتاً کمی از احتمالات را در نظر گرفت و توزیع احتمال مورد نظر را تخمین زد. در الگوریتم‌های نمونه‌برداری به جای اینکه توزیع احتمال در جهان‌های ممکن به عنوان مجموعه‌ای از اعداد نشان داده شود، از یک مجموعه مثال‌هایی از جهان‌های ممکن به نام نمونه‌ها (samples) استفاده می‌شود. به این صورت که مجموعه‌ای از نمونه‌ها تولید می‌شود و هر نمونه، خود یک جهان ممکن است و احتمال ایجاد یک جهان ممکن خاص باید برابر با احتمال آن جهان ممکن باشد. سپس می‌توان با استفاده از کسری از نمونه‌های برابر با آن جهان، احتمال یک جهان ممکن را تخمین زد. دسته‌بندی دیگری برای الگوریتم‌های استنتاج وجود دارد که به دو دسته تقسیم می‌شوند: دسته اول، الگوریتم‌های دقیق (exact) هستند که احتمال دقیق یک پرس‌وجو را با توجه به شواهد (evidence) تعریف شده در مدل به دست می‌آورند. دسته دوم، الگوریتم‌های تقریبی (approximate) هستند که این الگوریتم‌ها در بیشتر اوقات جوابی نزدیک به جواب صحیح پرس‌وجو را برمی‌گردانند.

۴–۱–۱– الگوریتم‌های استنتاج مبتنی بر فاکتور در برنامه‌نویسی Figaro

انواع الگوریتم‌های استنتاج مبتنی بر فاکتور موجود در برنامه‌نویسی فیگارو به این شرح می‌باشد: ۱–الگوریتم Variable Elimination (VE): یک الگوریتم مبتنی بر فاکتور دقیق می‌باشد. این الگوریتم با استفاده از عملیات جبری ساده متغیرهایی که در پرس‌وجو نیستند را حذف می‌کند. ترتیب حذف متغیرها در پیچیدگی این الگوریتم مؤثر می‌باشد. ۲–الگوریتم Belief Propagation (BP): یک الگوریتم مبتنی بر فاکتور تقریبی می‌باشد. مدل احتمالاتی مورد استفاده ممکن است مدل پیچیده‌ای

باشد و مسأله مورد نظر از متغیرهای زیادی تشکیل شده باشد در این موارد نمی‌توان از الگوریتم VE استفاده کرد و جواب دقیق را به دست آورد زیرا پیچیدگی این الگوریتم به صورت نمایی می‌شود در این‌گونه موارد از الگوریتم‌های تقریبی مانند Belief Propagation استفاده می‌شود. الگوریتم BP از عملیات ارسال پیام بین گره‌های شبکه استفاده می‌کند و تفاوت این الگوریتم با الگوریتم VE این است که این عملیات را به صورت هم‌زمان روی تمام متغیرهای شبکه انجام می‌دهد. الگوریتم BP را می‌توان بر روی هر تعداد تکرار اجرا کرد و بهترین جواب را بعد از آن تعداد تکرار دریافت کرد اما در حالت ایده‌آل هر چه تعداد تکرار بیشتر باشد جوابی که الگوریتم می‌دهد به جواب دقیق نزدیکتر است. در صورتی که الگوریتم BP خیلی کند عمل کند یا اینکه پاسخی که می‌دهد به پاسخ واقعی نزدیک نباشد از الگوریتم‌های نمونه‌برداری استفاده می‌شود. یک مبادله (trade-off) بین دقت و سرعت الگوریتم‌های استنتاج دقیق و تقریبی وجود دارد و نوع الگوریتم مورد استفاده بستگی به مدل احتمالاتی مسأله مورد نظر دارد.

۴–۱–۲– الگوریتم‌های استنتاج مبتنی بر نمونه‌برداری در برنامه‌نویسی Figaro

انواع الگوریتم‌های استنتاج مبتنی بر نمونه‌برداری موجود در برنامه‌نویسی فیگارو به این شرح می‌باشد: ۱–الگوریتم Importance Sampling: الگوریتم نمونه‌برداری اهمیت، هنگامی که با شرایطی که ارضا نمی‌شود مواجه شود، نمونه را رد می‌کند. اصل الگوریتم نمونه‌برداری اهمیت که آن را با الگوریتم‌های دیگر متمایز می‌کند این است که از نمونه‌های وزنی استفاده می‌کند و هر نمونه را با توجه به وزن نمونه در نظر می‌گیرد. در نمونه‌برداری اهمیت، هر نمونه با یک وزن همراه است که این وزن براساس مقادیر شرایط و محدودیت‌های نمونه می‌باشد. در نتیجه، توزیع احتمال در جهان‌های ممکن با وزن نمونه‌ها تعریف می‌شود. فرآیند انتخاب نمونه در این الگوریتم به صورت تصادفی است و احتمال انتخاب نمونه متناسب با وزن آن است. ۲–الگوریتم Markov Chain Monte Carlo (MCMC): در الگوریتم نمونه‌برداری اهمیت، تولید یک نمونه خوب که وزن بالایی دارد، می‌تواند مدت زمان زیادی طول بکشد و هم‌چنین بعد از تولید یک نمونه خوب، همه چیز دوباره با نمونه بعدی از ابتدا شروع می‌شود. دو مزیت عمده‌ی الگوریتم MCMC این است که: ۱– الگوریتم MCMC با سرعت بیشتری می‌تواند به نمونه‌هایی که احتمال بالایی دارند برسد و ۲– بعد از یافتن یک نمونه با احتمال زیاد، MCMC تمایل به ماندن در ناحیه با نمونه‌هایی با احتمال زیاد دارد. الگوریتم MCMC فیگارو از یک رویکرد استفاده می‌کند، که به عنوان الگوریتمی به نام Metropolis-Hastings (MH) معروف است که ایده اصلی این الگوریتم، استفاده از زنجیره‌ی مارکوف می‌باشد. الگوریتم MCMC دنباله‌ای از حالت‌ها را طی می‌کند و در هر مرحله از زمان، هر حالت را به‌طور تصادفی به حالت جدیدی انتقال می‌دهد. هر مرحله تصادفی است، اما گرایش به سمت حالت‌های با احتمال زیاد دارد. در پایان، الگوریتم معمولاً به حالت‌هایی می‌رسد که احتمال بالایی دارند. در این الگوریتم بعد از مدت‌ها انتقال، زمانی که توزیع در حالت فعلی نزدیک به توزیع واقعی است، در آن مرحله، یک نمونه ضبط می‌شود. فیگارو روش‌هایی برای ایجاد نمونه‌ای از الگوریتم MH با پیکربندی‌های مختلف از آرگومان‌های آن ارائه می‌دهد. متداول‌ترین روش، تعداد نمونه‌های مورد آزمایش، طرح پیشنهادی و اهداف پرس‌وجو را مشخص می‌کند.

۴–۲– الگوریتم‌های محاسبه مقادیر محتمل عناصر در Figaro

گاهی اوقات، به جای دانستن توزیع احتمال در مورد نتایج، نیاز هست تا مشخص شود که نتایج به چه صورت محتمل است، یعنی این‌که محتمل‌ترین حالت از خروجی که مقادیر با احتمال بالا از عناصر مدل را نشان دهد به چه صورت می‌باشد. پرس‌وجو که محتمل‌ترین حالت از متغیرهای مدل را نشان می‌دهد، به‌عنوان Most Probable Explanation (MPE) در فیگارو

شناخته می‌شود. فیگارو شامل سه الگوریتم برای محاسبه MPE است: دو الگوریتم فاکتور که شامل الگوریتم‌های MPEVariableElimination و MPEBeliefPropagation می‌باشد و یک الگوریتم نمونه‌برداری شامل MetropolisHastingsAnnealer می‌باشد. هر یک از این الگوریتم‌ها نمونه‌ای از الگوریتم MPE است. نکته اصلی که در مورد این الگوریتم‌ها وجود دارد این است که خواص آن‌ها شبیه به الگوریتم‌های معمولی استنتاج است و تنها تفاوت آن‌ها این است که به‌جای عملیات جمع از عملیات حداکثرسازی استفاده می‌شود. محاسبه محتمل‌ترین مقدار یک عنصر می‌تواند با استفاده از روش simulated annealing انجام شود که مبتنی بر الگوریتم Metropolis-Hastings است. ایده اصلی در مورد simulated annealing نمونه‌برداری از فضای مدل و ایجاد انتقال به حالت‌های با احتمال بالاتر در مدل است. در فیگارو، الگوریتم Metropolis-Hastings مبتنی بر simulated annealing بسیار شبیه به الگوریتم MH معمولی است.

۵–پارامترهای آزمایش مدل

در تکنیک‌های یادگیری ماشین نیاز به جمع‌آوری داده‌های آموزشی می‌باشد. داده‌های آموزشی برای تولید مراحل جدید بازی‌ها با استفاده از روش‌های یادگیری ماشین شامل تصاویری از مراحل بازی می‌باشد. در این مقاله برای تولید مراحل بازی Super Mario Bros با استفاده از مدل مارکوف چندبعدی، مجموعه داده‌های ورودی که شامل فایل‌های متنی از مراحل بازی می‌باشد، شامل ۷ مرحله از این بازی در نظر گرفته شده است که این مراحل به صورت دستی تعریف شده است. در مرحله‌ی نمونه‌برداری از مدل، مقدار پارامتر طول پیشروی (در الگوریتم با پارامتر l مشخص شده است) با مقادیر ۰ و ۲ مورد آزمایش قرار گرفته است. هم‌چنین طبق توضیحات داده شده در نمونه‌برداری مدل با استفاده از فیگارو در قسمت روش پیشنهادی، از سه الگوریتم استنتاج فیگارو در توابع نمونه‌برداری استفاده شده است که شامل الگوریتم‌های MPEVariableElimination و الگوریتم MPEBeliefPropagation و الگوریتم MetropolisHastingsAnnealer می‌باشد. در این آزمایشات، آرگومان ورودی الگوریتم MetropolisHastingsAnnealer که تعداد نمونه‌های جمع‌آوری شده را مشخص می‌کند، مقدار ۱۰۰۰ نمونه در نظر گرفته شده است و آرگومان‌های دیگر الگوریتم، مقادیر پیش فرض الگوریتم قرار داده شده است. آرگومان ورودی تعداد تکرار برای اجرا در الگوریتم BeliefPropagation، بعد از آزمایش روی مقادیر متفاوتی از تعداد تکرار، مقدار بهینه‌ی ۳۰ در نظر گرفته شده است.

۶–نتایج آزمایش روی مدل MdMC

در این پژوهش، مراحل تولید شده بازی‌ها بر اساس معیار میانگین حالت‌های نادیده (US) روی تعداد مراحل بازی نمونه‌برداری شده و زمان اجرای الگوریتم برای نمونه‌برداری یک مرحله از بازی ارزیابی می‌شوند. برای محاسبه‌ی میانگین حالت‌های نادیده، در آزمایشات انجام شده، ۱۰ مرحله از بازی Super Mario Bros نمونه‌برداری شده است و میانگین حالت‌های نادیده که در مراحل نمونه‌برداری شده رخ داده است، محاسبه شده است. جدول ۱ نتایج آزمایش مدل مارکوف چندبعدی را با پارامترهای آزمایش مدل که مشخص شده است نشان می‌دهد.

جدول ۱: نتایج آزمایش نمونه‌برداری مراحل بازی روی مدل MdMC

l	Unseen State (US)	Run time (sec)
۰	۴/۱	۶
۲	۲/۱	۷

هم‌چنین در این آزمایش، مدل احتمالاتی MdMC با استفاده از برنامه‌نویسی فیگارو و عناصر تعریف شده در فیگارو ایجاد شده است که نتایج آزمایش روی مدل MdMC با استفاده از برنامه‌نویسی فیگارو در جدول ۲ نشان داده شده است.

جدول ۲: نتایج آزمایش نمونه‌برداری مراحل بازی روی مدل MdMC با استفاده از برنامه‌نویسی فیگارو

Run time (sec)	Unseen State (US)	Algorithms	*l*
۱۳۲۰	۱	MPEVariableElimination	۰
۷۲۰	۴	MPEBeliefPropagation	
۱۰	۰/۹	MetropolisHastingsAnnealer	
۱۶۲۰	۱	MPEVariableElimination	۲
۱۰۸۰	۵	MPEBeliefPropagation	
۱۲	۰/۵	MetropolisHastingsAnnealer	

۷–ارزیابی نتایج و نتیجه‌گیری

همان‌طور که اهمیت تولید محتوای رویه‌ای به منظور خودکار نمودن فرآیند تولید محتوای بازی، در جهت کاهش هزینه طراحی و توسعه بازی‌ها افزایش می‌یابد، محققان راه‌های جدیدی را برای تولید محتوای با کیفیت در نظر می‌گیرند؛ این مقاله به رهیافت نوین استفاده از برنامه‌نویسی احتمالاتی در تولید محتوای رویه‌ای پرداخته است. در سال‌های اخیر زبان‌های برنامه‌نویسی احتمالاتی زیادی معرفی شده‌اند، که یکی از اهداف اصلی آن‌ها، ساخت مدل‌های احتمالی و پیاده‌سازی آن‌ها با استفاده از ترکیب احتمالات به‌همراه قدرت محاسباتی زبان‌های برنامه‌نویسی بوده است. در روش پیشنهادی این مقاله، تولید مراحل بازی Super Mario Bros مبتنی بر زنجیره‌های چندبعدی مارکوف، با استفاده از یکی از زبان‌های برنامه‌نویسی احتمالاتی به نام فیگارو انجام گرفته است.

با توجه به نتایج حاصل از آزمایش مدل احتمالاتی MdMC با استفاده از برنامه‌نویسی فیگارو (جدول ۲) و در مقایسه با نتایج موجود در آزمایش مدل MdMC بدون استفاده از برنامه‌نویسی احتمالاتی (جدول ۱) می‌توان نتیجه گرفت که پیاده‌سازی مدل احتمالاتی MdMC با استفاده از برنامه‌نویسی احتمالاتی فیگارو نتایج بهتری را با توجه به اینکه میانگین تعداد حالت‌های نادیده کمتری در مقایسه با نتایج جدول ۱ تولید شده است، دربرداشته است. این موضوع به این دلیل است که در برنامه‌نویسی فیگارو از الگوریتم‌های استنتاج فیگارو استفاده شده که این الگوریتم‌ها محتمل‌ترین مقدار برای هر موقعیت در نقشه خروجی را با توجه به داده‌های آموزش تولید می‌کنند و این موضوع باعث می‌شود تا تعداد حالت‌های نادیده تولید شده در هر موقعیت از نقشه بازی کاهش پیدا کند و یا حتی هرگز حالت‌های نادیده تولید نشود.

نتیجه‌ای دیگری که می‌توان از نتایج موجود در جدول ۲ دریافت، این است که با مقایسه‌ی مراحل تولید شده توسط دو الگوریتم استنتاج فیگارو می‌توان مشاهده کرد که الگوریتم MPEVariableElimination با توجه به اینکه میانگین حالت‌های نادیده کمتری در مقایسه با الگوریتم MPEBeliefPropagation تولید کرده است، عملکرد خوبی برای تولید خروجی داشته است. دلیل این موضوع این است که مدل احتمالاتی MdMC برای مسأله‌ی تولید مراحل بازی، یک مدل با عناصر دارای مقادیر گسسته و مشخص می‌باشد و همان‌طور که توضیح داده شد، الگوریتم VariableElimination یک الگوریتم دقیق است و

اگر استفاده از آن برای مسأله مورد نظر امکان‌پذیر باشد، نتیجه‌ی دقیقی را برای مدل تولید می‌کند. اما الگوریتم BeliefPropagation یک الگوریتم تقریبی است و تنها دلیل استفاده از آن، برای مسائل پیچیده که دارای متغیرهای با مقادیر پیوسته که در آن‌ها استفاده از الگوریتم VariableElimination امکان‌پذیر نیست، می‌باشد. هم‌چنین زمان اجرای الگوریتم MPEVariableElimination و الگوریتم MPEBeliefPropagation در آزمایش‌های انجام شده تقریباً به یک میزان بوده است. در نتیجه در مقایسه‌ی این دو الگوریتم، استفاده از الگوریتم MPEVariableElimination برای آزمایش‌ها دارای عملکرد بهتری می‌باشد.

در ادامه، سه الگوریتم موجود در جدول ۲ با یکدیگر مقایسه شده‌اند. الگوریتم نمونه‌گیری MetropolisHastingsAnnealer تعداد حالت‌های نادیده کمتری نسبت به دو الگوریتم دیگر تولید کرده است. هم‌چنین الگوریتم MetropolisHastingsAnnealer یک الگوریتم نمونه‌گیری است و توانسته خروجی‌های متفاوتی (تولید مراحل مختلف از بازی) را برای مدل مارکوف چندبعدی ایجاد کند. در نتیجه استفاده از الگوریتم MetropolisHastingsAnnealer نسبت به دو الگوریتم دیگر در آزمایش‌ها عملکرد بهتری داشته است. دلیل این موضوع این می‌باشد که الگوریتم MetropolisHastingsAnnealer در پی یافتن نمونه‌های خوب از مسأله (یعنی نمونه با احتمال رخداد بالا) می‌باشد. بنابراین، هنگامی که الگوریتم، نمونه با احتمال بالایی را یافت، تلاش می‌کند تا در آن ناحیه با نمونه های دارای احتمال رخداد بالا بماند یعنی مراحل بازی با احتمال رخداد بالا را نمونه‌برداری می‌کند در نتیجه استفاده از آن در مسأله‌ی مدلسازی مدل احتمالاتی MdMC نتیجه‌بخش واقع شده است.

همان‌طور که در جدول ۲ مشاهده می‌شود، متوسط زمان اجرای الگوریتم استنتاج MPEBeliefPropagation نسبت به متوسط زمان اجرای الگوریتم MPEVariableElimination برای اینکه یک مرحله از بازی را نمونه‌برداری کند کمی کوتاه‌تر می‌باشد. اما متوسط زمانی که الگوریتم استنتاج MetropolisHastingsAnnealer برای نمونه‌برداری یک مرحله از بازی سپری می‌کند، بسیار کوتاه‌تر از زمان اجرای این دو الگوریتم می‌باشد. این بدین معنی است که الگوریتم MetropolisHastingsAnnealer سریعتر از دو الگوریتم دیگر فیگارو که در جدول ۲ مشخص شده‌اند، می‌تواند نمونه‌های با احتمال بالا را جمع‌آوری کند و یک مرحله از بازی را نمونه‌برداری کند. در نتیجه اگر در این آزمایشات از الگوریتم MetropolisHastingsAnnealer برای نمونه برداری مراحل بازی استفاده شود، هم از نظر مدت زمان اجرا برای تولید خروجی، نزدیک به مدت زمان اجرا در نتایج جدول ۱ (نتایج آزمایش نمونه‌برداری مراحل بازی روی مدل MdMC) می‌باشد و هم این‌که مراحلی که توسط این الگوریتم نمونه‌برداری می‌شوند دارای میانگین تعداد حالت‌های نادیده کمتری نسبت به نتایج جدول ۱ می‌باشد. هم‌چنین در جدول ۲ تأثیر استفاده از پارامتر l (طول پیشروی) برای الگوریتم MetropolisHastingsAnnealer مشاهده می‌شود. به این صورت که تعداد حالت‌های نادیده تولید شده در مقدار ۲ برای پارامتر طول پیشروی نسبت به تعداد حالت‌های نادیده در مقدار صفر برای این پارامتر، کمتر بوده است یعنی با نمونه‌برداری چند کاشی جلوتر (دو کاشی جلوتر با در نظر گرفتن مقدار ۲ برای پارامتر l)، میانگین تعداد حالت‌های نادیده در مراحل نمونه‌برداری شده کاهش یافته است.

۸–پیشنهادها

همان‌طور که گفته شد، داده‌های آموزشی برای تولید مراحل جدید بازی‌ها با استفاده از روش‌های یادگیری ماشین شامل تصاویری از مراحل انواع بازی‌ها می‌باشد. اخیراً مجموعه‌ای از مراحل بازی‌های ویدئویی تحت عنوان Video Game Level

Corpus (VGLC) [۱۱]، ایجاد شده است که بازی‌ها در این مجموعه در سه قالب ارائه شده است. این مجموعه شامل ۴۲۸ مرحله از ۱۲ بازی می‌باشد که تصاویر مراحل بازی‌ها به همراه فایل متنی متناظر با آن‌ها در این مجموعه قرار دارد. در آزمایشاتی که در این مقاله انجام شده است، داده‌های موجود در پایگاه داده VGLC به دلیل در دسترس نبودن امکانات سخت‌افزاری مناسب برای اجرای الگوریتم‌های نمونه‌برداری روی این داده‌ها، مورد آزمایش قرار نگرفته است. به عنوان یک کار تحقیقاتی در آینده می‌توان الگوریتم‌های نمونه‌برداری را روی داده‌های موجود در پایگاه داده‌ی VGLC که یک پایگاه داده مناسب برای بازی‌های رایانه‌ای می‌باشد، مورد آزمایش قرار داد و نتایج تولید شده را با یک معیار ارزیابی مناسب با نام معیار قابلیت بازی‌پذیری (Playability) مورد ارزیابی قرار داد. این معیار، در واقع میزان قابلیت بازی‌پذیری برای یک مرحله از بازی را محاسبه می‌کند تا مشخص شود اشیا در بازی به شکل مناسب و در موقعیت‌های مناسب قرار گیرند. برای بازی Super Mario Bros می‌توان از Adam Summerville's A* agent [۱۲] برای تعیین Playability یک مرحله استفاده شود.

مراجع

1. *S. Snodgrass and S. Ontañón. "Learning to generate video game maps using markov models," IEEE Transactions on Computational Intelligence and AI in Games, pp. 410–422, 2017.*

2. *A. Summerville and M. Mateas,"Super Mario as a string: Platformer level generation via LSTMs", arXiv preprint arXiv:1603.00930, 2016.*

3. *M. Guzdial and M. O. Riedl,"Game Level Generation from Gameplay Videos," In AIIDE, 44–50, 2016.*

4. *V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. M. Smith, and S. Risi, "Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network," In Proceedings of the Genetic and Evolutionary Computation Conference, 221–228. ACM, 2018.*

5. *M. C. Fontaine,R. Liu,A. Khalifa, J. Modi, J. Togelius, A. K. Hoover, S. Nikolaidis, "Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network,"in Proceedings of the AAAI Conference on Artificial Intelligenc, 2021.*

6. *E., Maria, M. Jiang, and J. Togelius, "Search-based exploration and diagnosis of TOAD-GAN," in Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 17. no. 1. 2021.*

7. *https://en.wikipedia.org/wiki/Probabilistic_programming_language.*

۸. زهرا امیرجان و مرتضی درّی‌گیو، «تولید محتوای جدید بازی‌های رایانه‌ای با استفاده از برنامه‌سازی احتمالاتی»، ششمین کنفرانس بین‌المللی بازی‌های رایانه‌ای، فرصت‌ها و چالش‌ها، اصفهان، ایران، ۱۳۹۹.

9. *https://github.com/p2t2/figaro/tree/master/doc.*

10. *A. Pfeffer, "Practical probabilistic programming," Manning Publ., 2016.*

11. *A. J. Summerville, S. Snodgrass, M. Mateas, and Ontañón. "The VGLC: The video game level corpus," arXiv:1606.07487.*

12. *A. Summerville, S. Philip, and M. Mateas. Mcmcts pcg 4 smb: "Monte carlo tree search to guide platformer level generation," Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference, 2015.*