بسم الله الرحمن الرحیم

دانشگاه صنعتی شریف

دانشکده علوم ریاضی

پایان‌نامه

علوم کامپیوتر گرایش نظریه‌ی محاسبه

عنوان

مدلی برای بررسی و تحلیل یادگیری ماشین

نگارنده

علی ستاری جاوید

استاد راهنما

دکتر امیر دانشگر

دی ۱۳۹۵

اظهارنامه

(اصالت متن و محتوای پایان‌نامه‌ی کارشناسی ارشد)

عنوان پایان‌نامه: ____

____

نام استاد راهنما: ________ نام استاد راهنمای هم‌کار: ________ نام استاد مشاور: ________

این‌جانب ________ اظهار می‌دارم:

۱ - متن و نتایج علمی ارایه‌شده در این پایان‌نامه اصیل بوده و منحصراً توسط این‌جانب و زیر نظر استادان (راهنما، هم‌کار و مشاور) نام‌برده‌شده در بالا تهیه شده است.

۲ - متن پایان‌نامه به این صورت در هیچ جای دیگری منتشر نشده است.

۳ - متن و نتایج مندرج در این پایان‌نامه، حاصل تحقیقات این‌جانب به عنوان دانشجوی کارشناسی ارشد دانشگاه صنعتی شریف است.

۴ - کلیه‌ی مطالبی که از منابع دیگر در این پایان‌نامه مورد استفاده قرار گرفته، با ذکر مرجع مشخص شده است.

نام دانشجو: ____________

تاریخ:

امضاء:

نتایج تحقیقات مندرج در این پایان‌نامه و دستاوردهای مادی و معنوی ناشی از آن (شامل فرمول‌ها، نرم‌افزارها، سخت‌افزارها و مواردی که قابلیت ثبت اختراع دارد) متعلق به دانشگاه صنعتی شریف است. هیچ شخصیت حقیقی یا حقوقی بدون کسب اجازه از دانشگاه صنعتی شریف حق فروش و ادعای مالکیت مادی یا معنوی بر آن یا ثبت اختراع از آن را ندارد. همچنین کلیه‌ی حقوق مربوط به چاپ، تکثیر، نسخه‌برداری، ترجمه، اقتباس و نظایر آن در محیط‌های مختلف اعم از الکترونیکی، مجازی یا فیزیکی برای دانشگاه صنعتی شریف محفوظ است. نقل مطالب با ذکر ماخذ بلامانع است.

نام استادان راهنما: ____________ نام دانشجو: ____________

تاریخ: تاریخ:

امضاء: امضاء:

دانشگاه صنعتی شریف

دانشکده علوم ریاضی

پایان‌نامه

برای دریافت درجه کارشناسی‌ارشد در رشته

علوم کامپیوتر گرایش نظریهٔ محاسبه

عنوان: مدلی برای بررسی و تحلیل یادگیری ماشین

نگارش: علی ستاری جاوید

اعضای هیأت داوران:

دکتر امیر دانشگر(استاد راهنما)	امضاء: . . . .
دکتر محمد هادی فروغمند اعرابی	امضاء: . . . .
دکتر سید مجتبی مجتهدی	امضاء: . . . .

تاریخ: دی ۹۵

پیشگفتار

اواسط سال ۱۳۹۴ مطابق با پیشنهاد استاد ارجمند و گرامی، دکتر دانشگر، کار بر روی موضوع سلسه مراتب یادگیری ماشین را شروع کردم. شاید در ابتدا، بزرگی و سختی کار بر روی این ایده برایم کاملا شناخته شده نبود ولی در ادامه به این نتیجه رسیدم که موضوعی دشوار و کمتر بررسی شده را به عنوان پایان نامه دوره کارشناسی ارشد خود برگزیده‌ام.

در ابتدا هدف از این پژوهش بررسی سلسله مراتبی از ماشین‌های آموزش‌پذیر و در مجموع یادگیری ماشین بود. هرچند در ادامه موفق به معرفی مدلی برای توصیف این سلسله مراتب شدیم، ولی طبقه‌بندی الگوریتم‌های یادگیری در این مدل به صورت یکتا امکان‌پذیر نبود. به این معنی که یک الگوریتم را می‌شد در رده‌های مختلفی از این سلسله مراتب قرار داد. مدل ارائه شده برای توصیف این سلسه مراتب، نیز با هدف مشخص شدن سیر شکل گیری این اثر در ضمیمهٔآ آورده شده است. در ادامه با هدف تصحیح این مشکل، اندکی دایرهٔ پژوهش را گسترش دادیم و مدل‌های غیر متعارف دیگر محاسبه را بررسی کردیم. از جملهٔ این مدل‌ها ماشین‌های تعاملی[۱]، مدل محاسبه بر روی اعداد صحیح[۲] و ماشین‌های اینترنتی[۳] را می‌توان نام برد. در نتیجهٔ این جست و جو با مدل‌های دیگر محاسباتی مواجه شدیم که سعی در توصیف و تحلیل طبعات و محدودیت‌های یادگیری در سطح ماشین محاسبه‌گر داشتند. همچنین با راهنمایی‌های استاد داور، دکتر فروغمند، با مدل یادگیری تقریبا احتمالا درست[۴] آشنا شدم، هرچند این مدل ابزاری برای بررسی خواص همگرایی در الگوریتم‌های یادگیری را ارائه نمی‌داد. این در حالی بود که بسیاری از روش‌های یادگیری از همگرایی برای یافتن پاسخ مسائل یادگیری استفاده می‌کنند. در نتیجه مواجهه با این مدل‌ها و پرسش اصلی مطرح شده در این پایان‌نامه اندکی تغییر کرد:

آیا می‌توان مدلی جامع برای یادگیری ماشین ارائه داد که توانایی‌ها (و محدودیت) یادگیری ماشین در روش‌های مبتنی بر همگرایی را مستقل از الگوریتم مورد استفاده تحلیل کند؟

اعتقاد دارم این پژوهش تنها گامی نخستین در راستای پاسخ به پرسش مطرح شده است، و امید است دیگر پژوهش‌گران این راه را ادامه دهند.

نماد گذاری

در این پایان‌نامه، مجموعه‌ها با حروف تحریری بزرگ لاتین نوشته می‌شوند. مجموعه‌های خاص مانند $\mathbb{Z}$ و یا $\mathbb{R}$ از این قاعده مستثنی هستند و به صورت متداول نوشته می‌شوند. همچنین رشته‌ها، و اعداد بنا بر

محتوا با حروف کوچک لاتین نوشته شده‌اند. با توجه به اینکه هر ماشین محاسبه‌گر توسط یک رشته قابل معرفی است، ماشینی را که توسط رشتهٔ m معرفی می‌شود را به صورت $[m]$ نشان می‌دهیم. همچنین تابعی که توسط ماشین $[m]$ محاسبه می‌شود را به صورت ϕ_m نمایش می‌دهیم. در این نمایش (به غیر زمانی که مشخص شده باشد) تابع یک ورودی خواهد داشت.

پروردگارا

آنچه در این پایان‌نامه گردآوری شده، همه از الهامات تو بوده و ما در همه حال شکرگزار الطاف بیکرانت هستیم.

ادای احترام...

در این فرصت لازم می‌دانم از زحمات بی دریغ استاد گرانقدرم، جناب آقای دکتر دانشگر سپاس و قدردانی کنم. قطعا بدون کمک‌های بی بدیل ایشان، انجام این مهم امکان پذیر نبود.

در ادامه لازم می‌دانم از پدر، مادر و برادر عزیزم تشکر کنم، چرا که در تمام این مدت همواره حامی و پشتیبان من بوده‌اند.

و در نهایت از اساتید گرانقدر جناب آقای دکتر فروغمند و جناب آقای دکتر مجتهدی که زحمت مطالعه و داوری این پایان‌نامه را تقبل فرمودند، سپاس‌گزارم.

.

چکیده

مجموعه‌ای از الگوریتم‌ها تلاش می‌کنند مسائل را به صورت غیر مستقیم حل کنند. در این روش‌ها برنامه‌نویس خود روش مستقیمی برای حل مسئله ارائه نمی‌دهد، در مقابل برنامه خود تلاش می‌کند راه حل مسئله را بیابد. این روش‌ها در مدل یادگیری تقریبا احتمالا درست به صورت گسترده مورد بررسی قرار گرفته‌اند. هر چند این ساختار در میان محققین به عنوان مدل جامع برای توصیف یادگیری پذیرفته شده است، ولی برخی از خواص یادگیری ماشین را نمی‌تواند به خوبی توصیف کند. بسیاری از روش‌های یادگیری ماشین برای آموزش بر همگرایی تاکید دارند، این در حالی است که مدل PAC ابزاری برای بررسی سرعت و یا کیفیت همگرایی ارائه نمی‌دهد. در این مقاله ما ابتدا به بازگویی تعدادی مدل‌های محاسبه پرداخته‌ایم که به صورت مستقیم و یا غیر مستقیم در اجرای یک عملیات کاربرد دارند. سپس مدل یادگیری PAC را بر اساس مدل معرفی شده در کتاب [۴] بازگو کرده‌ایم و در نهایت به توصیف مدل یادگیری تدریجی پرداخته‌ایم. همچنین در این پایان‌نامه نشان داده‌ایم که هر مدل ارائه شده توسط ما در طبقه‌بندی مسائل آموزش‌پذیر، دقیقا مشابه با مدل یادگیری PAC رفتار می‌کند. لازم به ذکر است این پایان‌نامه به تحلیل الگوریتم‌های موجود در قالب مدل ارائه شده نمی‌پردازد و این بررسی‌ها را می‌توان در ادامه‌ی پژوهش حاضر انجام داد.

واژه‌های کلیدی: یادگیری ماشین، نظریهٔ محاسبه، مدل محاسبه، ماشین تورینگ

فهرست مطالب

فهرست جداول

فهرست تصاویر

فصل ۱
مقدمه

امروزه رایانه‌ها بخشی جدانشدنی از زندگی ما هستند. تلفن همراه، سیستم‌های بانکی، سیستم‌های ناوبری و غیره تنها بخش کوچکی از رایانه‌هایی هستند که زندگی امروز ما را امکان‌پذیر کرده‌اند. بدون شک این چنین استفادهٔ گسترده‌ای از رایانه‌ها نیازمند وجود و توسعهٔ تئوری‌ای بسیار قوی در مورد نحوهٔ عملکرد این ابزارهاست. با توجه به گستردگی کاربردها و ساختارهای دستگاه‌های محاسبه‌گر شاید بررسی ماشین‌های محاسبه‌گر در حالت کلی کاری بسیار دشوار و وسیع باشد. در نتیجه می‌توان یک ماشین را به صورت دستگاهی تصور کرد که عبارتی را به عنوان ورودی از کاربر دریافت می‌کند و پس از انجام محاسبه، خروجی را به صورت رشته‌ای از حروف تحویل می‌دهد.

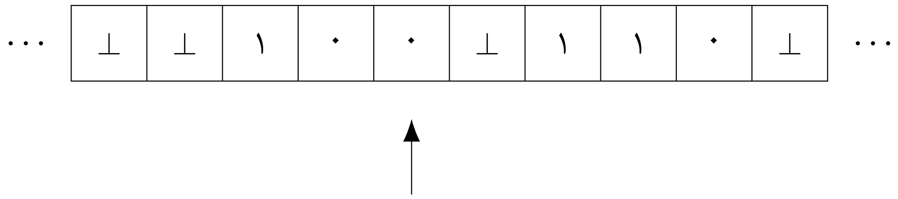
برای توصیف دقیق این گونه ماشین‌ها، تورینگ[۱] در سال ۱۹۳۶ ساختمانی مجرد معرفی نمود که توسط دیگر محققین «ماشین تورینگ»[۲] نام‌گذاری شد. این ساختار با وجود سادگی بسیار می‌تواند تمام محاسبات انجام قابل انجام توسط رایانه‌های امروزی، و یا حتی محسباتی فراتر از آن را بیان کند. مدل‌های محاسبهٔ دیگری نیز در راستای بررسی محاسبه‌پذیری به وجود آمده‌اند که می‌توان از جملهٔ آنها به مدل «توابع بازگشتی»[۳] که در سال ۱۹۳۳ توسط گودل[۴] ارائه شد، و یا «حساب لاندا»[۵] که در سال ۱۹۳۶ توسط چرچ[۶] معرفی شد نیز اشاره کرد.

با وجود تنوع بسیار زیاد مدل‌های محاسبه به تدریج نشان داده شد که این مدل‌ها از توان محاسباتی یکسانی برخوردارند و مجموعهٔ یکسانی را به عنوان توابع محاسبه‌پذیر معرفی می‌کنند. در نتیجه‌ی این مشاهدات فرضیه معروف چرچ-تورینگ مطرح شد که ادعا می‌کند تابعی «محاسبه‌پذیر»[۷] است اگر و تنها اگر توسط ماشین تورینگ قابل محاسبه باشد. البته صورت این گزاره به هیچ وجه از دقت کافی برخوردار نیست، چرا که کلمهٔ محاسبه‌پذیر تعریفی انتزاعی دارد. همچنین می‌توان این ایراد را به تز چرچ-تورینگ وارد نمود که محاسبه‌پذیری صفتی نسبی است و نباید آن را به صورت مطلق تعریف نمود. به عنوان نمونه، دستگاهی با محدودیت حافظه توانایی حل بسیاری از مسائل را نخواهد داشت؛ و یا اگر دستگاهی که به دانشی فرای توابع محاسبه‌پذیر دسترسی داشته باشد، می‌تواند حجم بسیار گسترده‌تری از مسائل را حل کند.

برای تصحیح ایرادات بالا در ادبیات نظریهٔ محاسبه تعریفی ارائه شده که در بسیاری موارد می‌تواند راهگشا باشد. «تابعی محاسبه‌پذیر است که یک ریاضی‌دان دقیق بتواند با پیروی از مجموعه دستوراتی ساده آن را در زمانی متناهی محاسبه کند.» هر چند در این تعریف نیز کلمهٔ «ساده» می‌تواند مورد نقد و بررسی قرار بگیرد.

دیگر ایراداتی که به این تز وارد می‌شود ساختار محاسبه‌ای است که یک ریاضی‌دان در واقعیت پیش

[۱] Alan Turing
[۲] Turing Machine
[۳] Recursive Functions
[۴] Kurt Gödel
[۵] Lambda Calculus
[۶] Alonzo Church
[۷] Computable



شکل ۱.۱: نمایشی از وضعیت یک ماشین تورینگ

می‌گیرد. بر خلاف آنچه در محاسبه توسط ماشین تورینگ فرض می‌شود، محاسباتی که در واقعیت رخ می‌دهند به صورت یک جعبهٔ سیاه و کاملا مستقل از محیط پیش نمی‌روند و معمولا محاسبه‌گر در طول محاسبه با محیط اطراف «تعامل»[۸] دارد. همچنین بر خلاف مدل مجرد معرفی شده، در دنیای خارج، انجام یک محاسبه با مقادیری از ورودی‌ها معمولا بر محاسبات بعدی تاثیرگذار خواهد بود و ممکن است باعث تغییر در روند، زمان و یا حتی نتیجهٔ محاسبات آتی گردد.

در ادامه ابتدا به بررسی توصیف کلاسیک یک ماشین تورینگ می‌پردازیم و سپس برخی روش‌های ارائه شده در راستای تصحیح مشکلات ذکر شده را بیان می‌کنیم.

۱.۱ توصیف کلی یک ماشین تورینگ ساده

ماشین تورینگ یک دستگاه انتزاعی است که توسط آلن تورینگ در سال ۱۹۳۶ ابداع شد. برخلاف ساختار بسیار ساده، بر اساس تز چرچ-تورینگ این ماشین توانایی شبیه‌سازی تمام الگوریتم‌ها را داراست. همین امر به ما اجازه می‌دهد با ساخت یک ماشین تورینگ (در حقیقت یک رایانه) رفتار هر ماشین دیگری را شبیه‌سازی کنیم. هر چند ماشین تورینگ به شیوه‌های مختلفی بیان شده است، در این پایان‌نامه ما به ذکر یکی از این انواع بسنده می‌کنیم. لازم به ذکر است که می‌توان نشان داد در عمل توان محاسباتی بسیاری از بیان‌ها با یکدیگر برابر است.

۱.۱.۱ تعریف

یک ماشین تورینگ توسط یک «تابع گذر»[۹] ($\delta : \Sigma \cup \{\perp\} \times \mathcal{S} \to \Sigma \cup \{\perp\} \times \mathcal{S} \times \{l, r\}$) و مجموعه‌ای متناهی به نام مجموعه حالات ($\mathcal{S}$) توصیف می‌شود. از مجموعهٔ حالات $\mathcal{S}$ یک حالت به عنوان حالت ابتدایی ماشین مشخص شده است، به این معنی که زمانی که ماشین محاسبهٔ خود را شروع می‌کند در حالت ابتدایی قرار دارد. همچنین ماشین برای انجام محاسبه مجهز به یک نوار حافظه به طول نامتناهی است که در هر خانه‌ای یکی از حروف الفبای ماشین (Σ) یا نماد مشخص تهی ($\perp$) می‌تواند قرار بگیرد. ضمنا ماشین در هر مرحله از عملیات خود، یکی از اعضای مجموعهٔ حالات را به عنوان حالت فعلی به صورت جداگانه نگهداری می‌کند. در نهایت ماشین دارای اشاره‌گری است که یکی از خانه‌های حافظه را مشخص می‌کند. در ابتدای

[۸] Interaction
[۹] Transition Function

ورودی		خروجی		
حالت	حرف	حالت	حرف	حرکت
true	۰	false	$\perp$	r
false	۰	true	$\perp$	r
true	۱	true	$\perp$	r
false	۱	false	$\perp$	r
true	$\perp$	end	۱	r
false	$\perp$	end	۰	r

جدول ۱.۱: دستورات ماشین پذیرندهٔ عبارات با تعداد زوجی ۰

کار، ورودی ماشین در «نوار حافظه»[۱۰] نوشته می‌شود و اشاره‌گر ماشین به اولین خانهٔ ورودی اشاره می‌کند. همچنین در دیگر خانه‌های نوار حافظه مقدار ثابت تهی قرار می‌گیرد.

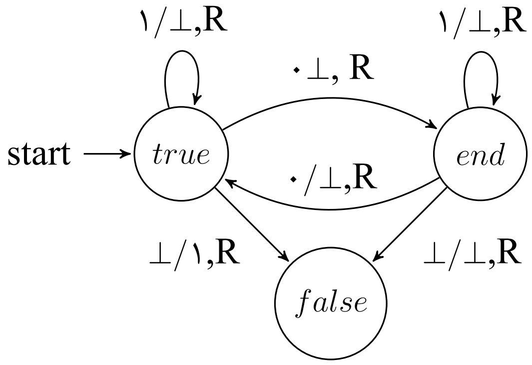
اساس کار محاسبه در یک ماشین تورینگ بر مبنای خواندن محتوای نوشته شده در نوار، جایگزین نمودن آن با محتوای جدید و حرکت اشاره‌گر است. در هر مرحله از محاسبه، ماشین ابتدا حرف مشخص شده توسط «اشاره‌گر»[۱۱] را از روی نوار می‌خواند و سپس با توجه به حالت فعلی‌اش مقدار تابع گذر را محاسبه می‌کند. اگر تابع گذر به ازای زوج مرتب حالت و حرف مورد نظر تعریف نشده باشد، ماشین متوقف می‌شود در غیر این صورت حالت داخلی ماشین تغییر می‌کند و مقدار جدید را در نوار حافظه نوشته می‌شود. در نهایت با توجه به جهت گذر اشاره‌گر را یک خانه به چپ و یا راست حرکت می‌دهد. شکل ۱.۱ نمایی از وضعیت یک ماشین در حال انجام محاسبه را نشان می‌دهد.

تعریف ۱.۱. به مجموعهٔ نوار، حالت فعلی ماشین، و مکان اشاره‌گر، «پیکربندی»[۱۲] گفته می‌شود. به عبارت دیگر پیکربندی یک ماشین شامل تمام اطلاعاتی است که برای شبیه‌سازی وضعیت عملکرد ماشین باید نگه‌داری شود.

۲.۱.۱ مثال

همان‌طور که گفته شد ماشین‌های تورینگ توانایی حل بسیاری از مسائل را دارا هستند. به عنوان مثال در صورت اجرای ماشینی با تابع گذری مانند جدول ۱.۱ و با حالت اولیهٔ true بر روی رشته‌های مختلف از الفبای $\{۰, ۱\}$ ، اگر عبارت تعداد زوجی ۰ داشته باشد پس از اتمام محاسبات عبارت ۱ چاپ خواهد شد و در غیر این صورت مقدار ۰ قرار خواهد گرفت. ماشین دیگری با همین تابع گذر اگر از حالت false محاسباتش را شروع می‌نمود دقیقا خروجی‌هایی عکس خروجی‌های ذکر شده تولید می‌نمود. همین ماشین را می‌توان به صورت یک گراف نمایش داد، به این ترتیب که رئوس گراف نشان‌دهندهٔ حالت‌های ماشین باشند و یال‌ها تابع گذر را مشخص کنند. گراف متناظر با ماشین معرفی شده در شکل ۱.۲ نمایش داده شده است.

[۱۰] Memory Tape
[۱۱] Head
[۱۲] Confguration



شکل ۱.۲: نمایش ماشین تورینگ به صورت گراف

۱.۱.۳ ماشین جهانی

یکی از جالب‌ترین و مهم‌ترین خواص ماشین ارائه شده توسط تورینگ وجود «ماشین جهانی»[۱۳] است. همان‌طور که گفته شد یک ماشین تورینگ توسط مجموعهٔ حالات، حالت اولیه و تابع گذر معرفی می‌شود. اما طبق تعریف مجموعهٔ حالات و مجموعهٔ حروف ماشین تورینگ، خود متناهی هستند. در نتیجه تابع گذر نیز نمی‌تواند بیش از متناهی دستور را شامل شود. در نهایت می‌توان نتیجه گرفت که یک ماشین تورینگ را می‌توان، مستقل از الفبای «کدگذاری»[۱۴]، با متناهی داده به صورت یکتا مشخص کرد. حال ماشینی را تصور کنید که ورودی‌ای شامل دو بخش دریافت می‌کند: به طوری که بخش اول توصیف یک ماشین تورینگ و بخش دوم عبارتی دلخواه باشد. این ماشین می‌تواند رفتار ماشین توصیف شده را شبیه‌سازی کند به این ترتیب که خروجی‌ای مشابه خروجی آن ماشین تولید کند. به این چنین دستگاهی، ماشین جهانی گفته می‌شود. می‌توان نشان داد که یک ماشین تورینگ جهانی برای شبیه‌سازی محاسبهٔ هر ماشین دیگری وجود دارد.

۱.۱.۴ محدودیت‌ها

هر چند ماشین محاسبهٔ تورینگ از توانایی‌های بسیار بالایی برخوردار است، اما می‌توان به سادگی نشان داد مسائلی وجود دارند که توسط ماشین تورینگ قابل حل نیستند. همین‌طور می‌توان نشان داد گروه دیگری از مسائل وجود دارند که هر چند ماشین می‌تواند پاسخ آن‌ها را محاسبه کند، اما برای این کار به زمانی بسیار طولانی نیازمند است.

همان‌طور که گفته شد می‌توان یک ماشین تورینگ را توسط متناهی داده توصیف نمود. متناهی بودن داده‌های مربوط به هر ماشین تورینگ باعث می‌گردد در مجموع تعداد ماشین‌های تورینگ شمارا باشد. این در حالی است که تعداد توابع از رشته‌های روی یک الفبا، به رشته‌هایی روی همان الفبا ناشمارا است. در نتیجه قطعا تابعی وجود دارد که توسط هیچ ماشینی نمی‌توان آن را محاسبه نمود.

[۱۳] Universal Machine
[۱۴] Coding

مسئلهٔ توقف

این استدلال، یکی از ابتدایی‌ترین اثبات‌ها برای ناتوانی ماشین تورینگ در پاسخ‌گویی به برخی مسائل بود. هر چند اثبات ذکر شده هیچ تابع محاسبه‌ناپذیری را معرفی نمی‌کند، اما دلیل کافی است که به دنبال این چنین تابعی بگردیم. یکی از توابعی که چنین خاصیتی دارد و به «مسئلهٔ توقف»[۱۵] مشهور است را می‌توان به صورت زیر بیان کرد:

> یک ماشین تورینگ را می‌توان توسط عبارتی در الفبای $\{۰, ۱\}$ معرفی نمود. همچنین می‌توان در کنار عبارت معرف یک ماشین تورینگ، می‌توان یک عبارت ثانویه به عنوان ورودی برای ماشین مشخص شده قرار داد. حال آیا ماشینی که توسط عبارت داده شده مشخص می‌شود، با گرفتن ورودی داده شده، محاسبه را کامل انجام می‌دهد، و یا در حلقه‌ای نامتناهی گیر می‌کند؟

اثبات محاسبه‌ناپذیر بودن این مسئله از حوصلهٔ این متن خارج است، هر چند می‌توانید برای بررسی دقیق‌تر این مسئله به کتاب [۵، بخش ۱.۵.۱] مراجعه نمایید.

پیچیدگی محاسبه

ماشین تورینگ در صورت وجود محدودیت در زمان محاسبه و یا حافظهٔ مورد استفاده ممکن است نتواند برخی مسائل را پاسخ دهد. به حداقل امکانات مورد نیاز ماشین برای انجام یک محاسبه «پیچیدگی»[۱۶] آن محاسبه گفته می‌شود. با اندکی ساده‌سازی صورت مسئلهٔ توقف می‌توان آن را به مسئله‌ای قابل حل تبدیل کرد. فرض کنید تابع خواسته شده علاوه بر توصیف ماشین مورد بحث، تعداد گذرهایی که ماشین مجاز است قبل از نوشتن پاسخ بپیماید را نیز ورودی بگیرد. با توجه به محدود شدن زمان عملکرد ماشین، همواره می‌توان رفتار آن را توسط یک ماشین جهانی شبیه سازی کرد به طوری که ماشین جهانی پس از اتمام زمان محاسبه و یا اتمام محاسبهٔ اشین شبیه‌سازی شده، خروجی را برآورد کند. اما برای این کار به زمانی بسیار طولانی نیاز خواهد داشت. با توجه به اینکه محدودیت زمانی می‌تواند در مبنای دو ورودی داده شود ممکن است زمان کار ماشین جهانی از مرتبهٔ نمایی از طول ورودی باشد. همچنین می‌توان ثابت نمود که این مسئله راه حلی سریع‌تری نیز ندارد و هر راه حل دیگری به اندازهٔ شبیه‌سازی رفتار ماشین زمان‌بر خواهد بود.

۱.۱.۵ دیگر توصیفات

یکی از دلایلی که منجر به طرح فرضیهٔ چرچ-تورینگ شد، وجود مدل‌های متعدد (و متفاوت) محاسبه بود که مجموعه‌ای یکسان را به عنوان توابع محاسبه‌پذیر معرفی می‌کردند. به غیر از تعریف حاضر از ماشین

[۱۵] Halting Problem
[۱۶] Complexity

تورینگ تعاریف دیگری نیز ارائه شده‌اند که همگی از نظر توان محاسباتی با یکدیگر برابری می‌کنند.

ماشین تورینگ با چندین نوار حافظه

یکی انواع تغییراتی که در ماشین تورینگ می‌توان داد اضافه کردن چندین نوار حافظه به ماشین است. می‌توان نشان داد حتی اگر یک ماشین دسترسی به بیش از یک نوار حافظه داشته باشد و در هر مرحله از عملکرد خود بتواند به اطلاعات موجود در نوارها به طور همزمان دسترسی داشته باشد و یا آن‌ها را تغییر دهد، تفاوتی در کارایی ماشین ایجاد نخواهد شد. به عنوان مثال در ماشینی که از دو نوار حافظه بهره می‌برد، تابع گذر به صورت $\delta : \Sigma \cup \{\perp\} \times \Sigma \cup \{\perp\} \times \mathcal{S} \rightarrow \Sigma \cup \{\perp\} \times \Sigma \cup \{\perp\} \times \mathcal{S} \times \{l, r\} \times \{l, r\}$ خواهد بود. این تابع مقدار موجود در نوارهای حافظه را به صورت جداگانه ورودی می‌گیرد و جهت حرکت هر کدام از اشاره‌گرها را به صورت جداگانه مشخص می‌کند.

ماشین‌های تورینگ غیر قطعی

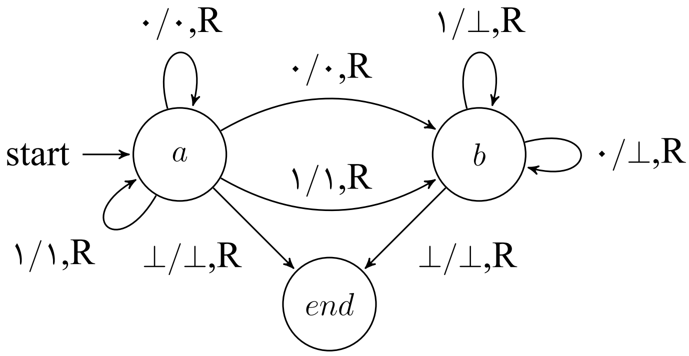
از دیگر تغییراتی که می‌شود در ساختار ماشین تورینگ ایجاد نمود اضافه کردن گذرهای متفاوت به ازای یک وضعیت مشابه است. به عبارت دیگر تابع گذر را به رابطه‌ای میان وضعیت ماشین و گذرهای احتمالی گسترش دهیم. به این ترتیب ممکن است ماشین تورینگ در هر یک از مراحل محاسبهٔ خود با بیش از یک مسیر برای پردازش مواجه شود. با توجه به انتخاب‌هایی که «ماشین غیر قطعی»[۱۷] در طول پردازش خود انجام می‌دهد، این احتمال وجود دارد که خروجی ماشین منحصر به یک جواب یکتا نباشد، و یا حتی در بعضی از مسیرهای پردازش، محاسبهٔ ماشین تمام نشود در حالی که در بعضی مسیرهای دیگر پردازش به درستی کامل شود. در ماشین‌های غیر قطعی یکتا نبودن جواب خروجی در واقع به ما کمک می‌کند که بتوانیم در میان حجم عظیم داده‌هایی محتمل خروجی به دنبال یک خروجی خاص بگردیم. به عبارت دیگر می‌گوییم ماشین غیر قطعی خروجی‌ای مانند x را چاپ می‌کند اگر و تنها اگر حداقل در انتهای یکی از مسیرهای پردازش نتیجهٔ حاصل برابر x باشد.

به عنوان مثال شکل ۱.۳ یک ماشین تورینگ را معرفی می‌کند که به ازای هر ورودی، ممکن است هر کدام زیر رشته‌هایی که از ابتدای ورودی شروع می‌شوند را خروجی دهد.

ماشین تورینگ احتمالاتی

یک الگوریتم احتمالاتی الگوریتمی‌ست که در طول مراحل محاسبهٔ خود برخی تصمیمات را به صورت تصادفی می‌گیرد. به عنوان مثال این تصمیم می‌تواند انتخاب یک عدد صحیح در ابتدای شروع به کار الگوریتم باشد. این الگوریتم‌های برای اجرا شدن نیاز به یک مولد عدد تصادفی دارند، که این مولد در ساختار یک ماشین تورینگ معمولی وجود ندارد. می‌توان به سادگی نشان داد که مولد عدد تصادفی

[۱۷] Nondeterministic Machine



شکل ۱.۳: یک ماشین تورینگ غیر قطعی

کافی‌ست تنها یک حرف از الفبا را به صورت تصادفی انتخاب کند، و می‌توان با تولید حروف تصادفی به دفعات متعدد، می‌توان رشته‌های به دلخواه طولانی با توزیعی معین تولید نمود.

«ماشین احتمالاتی»[۱۸] از نظر ساختاری مشابه ماشین‌های تورینگ غیر قطعی هستند، ولی در نحوهٔ عملکرد و تعبیری که از پاسخ می‌شود با یکدیگر تفاوت دارند. یک ماشین تورینگ احتمالاتی ممکن است به ازای یک ورودی چندین خروجی مختلف ارائه دهد و هر خروجی را با احتمال مشخص و قابل محاسبه‌ای چاپ کند. این تعبیر با آنچه در مورد ماشین‌های تورینگ غیر قطعی گفته شد متفاوت است چرا که در مورد ماشین تورینگ غیر قطعی ما فقط به دنبال یک جواب منحصر به فرد بودیم و بقیهٔ جواب‌های که تولید می‌شدند را نادیده می‌گرفتیم. این در حالی است که در ماشین‌های تورینگ احتمالاتی تمام جواب‌هایی که تولید می‌شوند معتبر و تاثیرگذاراند. به عبارت دیگر، بر خلاف دیگر ماشین‌هایی که تا به این‌جا معرفی شده‌اند، احتمال رخداد هر کدام از خروجی‌های ماشین موضوع بررسی در این ماشین‌هاست.

۱.۱.۶ ماشین‌های تورینگ دارای پیشگو

مقایسه کردن دو مسئله از نظر پیچیدگی مستقیما با استفاده از تعریف ماشین تورینگ کاری بسیار دشوار است. یکی از راه حل‌ها برای این مشکل، اضافه کردن «پیشگو»[۱۹] به تعریف ماشین تورینگ است.

تعریف

یک ماشین تورینگ دارای پیشگو از لحاظ ساختاری همانند یک ماشین تورینگ ساده با دو نوار حافظه است، که نوار دوم را نوار پیشگو می‌نامیم. همچنین ماشین به تابعی مانند o به نام تابع پیشگو دسترسی دارد. در یک ماشین دارای پیشگو، دو حالت خاص نیز به نام‌های حالت «پرسش» و حالت «پاسخ» وجود دارد. زمانی که ماشین وارد حالت «پرسش» می‌شود تابع o بر روی عبارت موجود در نوار پیشگو اجرا می‌شود و مقدار تابع در نوار جایگزین می‌گردد. سپس ماشین به حالت «پاسخ» می‌رود و ادامهٔ عملیات ماشین مطابق روند معمول ماشین تورینگ دنبال می‌شود.

[۱۸] Probabilistic Machine
[۱۹] Oracle

مثال

تحویل مسئلهٔ چاپ به مسئلهٔ توقف

همانند مسئلهٔ توقف مسائل بسیار دیگری وجود دارد که یک ماشین تورینگ ساده نمی‌تواند پاسخ آنها را محاسبه کند. از جملهٔ این مسائل می‌توان به «مسئلهٔ چاپ»[۲۰] اشاره کرد. صورت این مسئله به شرح زیر است:

> یک ماشین تورینگ را می‌توان توسط عبارتی در الفبای $\{0,1\}$ معرفی نمود. حال آیا ماشینی که توسط عبارت داده شده مشخص می‌شود، می‌تواند به ازای یک ورودی خاص، رشتهٔ تک حرفی «۰» را چاپ می‌کند یا خیر؟ اگر جواب مثبت است، خروجی تابع عبارت تک حرفی «۰» است. در غیر این صورت تابع عبارت «۱» را باید خروجی دهد.

با اینکه مسئله چاپ به ظاهر ساختاری بسیار پیچیده‌تری نسبت به مسئلهٔ توقف دارد ولی با کمک ماشین‌های دارای پیشگو می‌توان نشان داد هر دو این مسائل از یک میزان سختی برخوردارند. به عبارت دیگر اگر بتوانیم الگوریتمی برای محاسبهٔ پاسخ به یکی از این مسائل را ارائه دهیم، می‌توان برای مسئلهٔ دیگر نیز الگوریتمی ارائه کرد. به این روش اثبات «تحویل»[۲۱] گفته می‌شود.

تحویل در مسائل پیچیدگی محاسبه

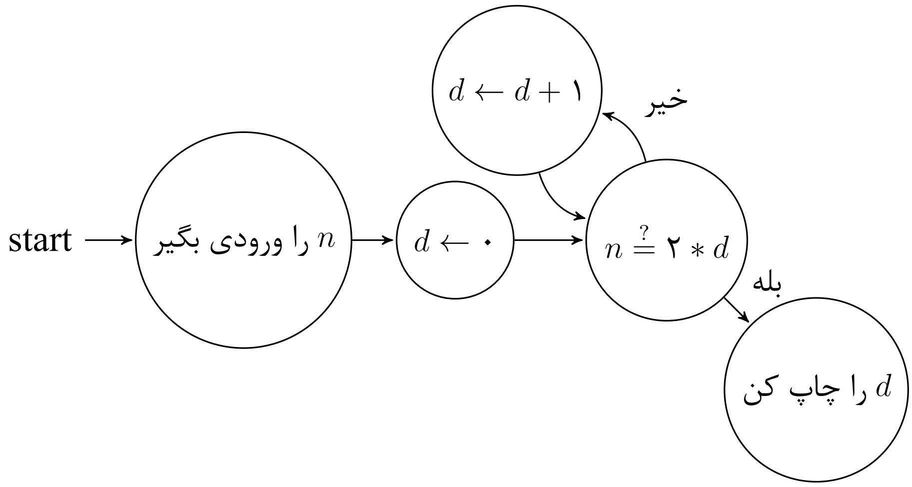
همان‌طور که گفته شد برای حل بعضی مسائل تعداد اعمال زیادی باید انجام شود محاسبه را نمی‌توان به سرعت پایان داد. بررسی پیچیدگی مسائل مختلف در مقایسه با یکدیگر نیز از جمله سوال‌هایی است که می‌توان با استفاده از ماشین‌های دارای پیشگو به آن پاسخ داد.

۱.۲ مجموعه‌های محاسبه‌پذیر

طبق آنچه گفته شد، یک ماشین می‌تواند فقط بر روی زیرمجموعه‌ای از ورودی‌های ممکن جواب چاپ کند و در بقیه‌ی حالات متوقف نشود. به زیر مجموعه‌ای از ورودی‌های یک ماشین مانند $\lfloor m \rfloor$ که ماشین بر روی آن‌ها توقف می‌کند «مجموعهٔ کار»[22] ماشین گفته می‌شود و آن را با $W_{\lfloor m \rfloor}$ نشان می‌دهند.

برای بعضی از ماشین‌ها شناختن مجموعه‌ی کار، قابل محاسبه است. به این معنا که می‌توان با گرفتن عضوی از فضای ورودی‌های ممکن $\lfloor m \rfloor$، تشخیص داد که آیا این عضو در مجموعهٔ $W_{\lfloor m \rfloor}$ وجود دارد یا خیر؟ به عنوان مثال ماشینی را در نظر بگیرید که یک عدد صحیح مانند n را به عنوان ورودی دریافت می‌کند، سپس اگر عدد زوج بود، مقدار $n/2$ را چاپ می‌کند، در غیر این صورت هیچ وقت متوقف نمی‌شود. توصیف غیر دقیق روش عملکرد این ماشین می‌توانند همانند شکل ۱.۴ باشد. تشخیص اینکه یک داده جزو

[۲۰] Printing Problem
[۲۱] Reduction
[۲۲] Halting Set



شکل ۱.۴: ماشین تقسیم کننده

مجموعهٔ کار این ماشین هست یا خیر کار ساده‌ای است. البته این نمودار توصیفی کلی و نادقیق از یک ماشین است ولی می‌توان ماشین تورینگی با این عملکرد را معرفی کرد.

با توجه به ساختار ماشین فوق می‌توان بدون اجرا کردن خود ماشین به سادگی تشخیص داد که آیا ماشین برای ورودی‌ای مانند x توقف می‌کند یا خیر؟ کافی است تشخیص دهیم آیا ورودی زوج است یا فرد؟

اگر اعضای یک مجموعه را بتوان به طور دقیق مشخص نمود، و بتوانیم تعلق و یا عدم تعلق یک عضو به مجموعه را توسط یک ماشین تورینگ تشخیص دهیم به آن مجموعه، یک «مجموعهٔ بازگشتی»[۲۳] می‌گویم.

در بعضی موارد می‌توان عضو بودن یک عنصر خاص در مجموعه را بررسی نمود، ولی عدم عضویت توسط هیچ ماشین تورینگی قابل اثبات نیست. به عنوان مثال مجموعه‌ای را در نظر بگیرید که اعضای آن همگی کدهای ماشین تورینگ باشند. به طور خاص ماشین‌های تورینگی که با ورودی گرفتن کد خودشان توقف می‌کنند. به بیان دقیق‌تر فرض کنید:

$$\mathcal{K} = \{x \in \Sigma^* | \lfloor x \rfloor (x) \downarrow\}$$

در اینجا علامت $\downarrow$ به معنای توقف ماشین پس از انجام محاسبه بر روی ورودی داده شده است. اگر ماشین بر روی یک ورودی توقف ننماید، عدم توقف را با علامت $\uparrow$ نشان می‌دهیم.

حال با توجه به آن‌چه در مورد مسئلهٔ توقف گفته شد، هیچ ماشینی وجود ندارد که بتواند مجموعهٔ $\mathcal{K}$ را معرفی کند. البته می‌توانیم با استفاده از ماشین تورینگ جهانی عضویت در مجموعهٔ $\mathcal{K}$ را بررسی نماییم، ولی عدم عضویت یک عبارت در مجموعهٔ $\mathcal{K}$ قابل اثبات نیست. به این چنین مجموعه‌هایی «مجموعهٔ شمارشی بازگشتی»[۲۴] گفته می‌شود.

[۲۳] Recursive
[۲۴] Recursively Enumerable

۱.۳ مروری بر فصل‌ها

در فصل اول مقدمه‌ای از اصول و قضایای اصلی نظریهٔ محاسبه گفته شد، تعدادی از ساختارهای ابتدایی را معرفی کردیم و برخی قضایای اساسی را معرفی نمودیم. منبع اصلی استفاده شده در این فصل کتاب [۵] بوده است. در ادامه با توجه به اهمیت حد ریاضی در یادگیری ماشین در فصل دوم نمونه‌ای از ماشین‌های با قابلیت پردازش بر روی اعداد حقیقی معرفی می‌نماییم. مدل معرفی شده در این فصل بر اساس مقالهٔ [۲] نوشته شده است. بسیاری از الگوریتم‌های آموزشگر، به خصوص روش‌های بدون ناظر، در مراحل آموزش خود نیازمند تعامل با محیط هستند، در نتیجه در فصل سوم به معرفی ماشین‌های تعاملی می‌پردازیم. برای نوشتن فصل سوم از دو مقالهٔ [۳] و [۱] استفاده شده است. سپس در فصل چهارم به معرفی مدل‌هایی می‌پردازیم که سعی کرده‌اند یادگیری ماشین را از نظر محاسباتی بررسی کنند. در این فصل مدلی تحت عنوان ماشین تورینگ عصبی [۶] معرفی، و همچنین با استفاده از [۷] مدلی برای بررسی الگوریتم‌های تکاملی ارائه شده است. در نهایت در فصل پنجم مدلی برای بررسی توانایی‌ها و محدودیت‌های محاسباتی الگوریتم‌های آموزش پذیر ارائه می‌دهیم و نشان می‌دهیم این مدل قابلیت توصیف الگوریتم‌های یادگیری گفته شده را داراست.

فصل ۲
محاسبه بر روی اعداد حقیقی

هر چند تعامل یکی از کمبودهای اساسی مدل‌های محاسبهٔ فعلی‌ست ولی تنها مشکلی نیست که ما برای توصیف الگوریتم‌ها با آن برخورد کرده‌ایم. با اینکه نظریهٔ کلاسیک محاسبه توانسته است بسیاری از نیازهای ما را در تحلیل و توصیف الگوریتم‌ها و فرآیند محاسبه برآورده کند، ولی وابستگی آن به الفبای $\{۰, ۱\}$ و یا در حالت کلی‌تر الفبای متناهی باعث می‌شود نتوانیم بسیاری فرایندها را به خوبی تحلیل کنیم. «محاسبات علمی»[۱]، که توسط نیوتون پایه‌گذاری شده‌اند، در بسیاری موارد مبتنی بر دقت اعداد حقیقی هستند.

هر چند معمولا می‌توان بسیاری از این الگوریتم‌ها را به کمک روش‌های عددی به صورت تقریبی محاسبه نمود ولی در بعضی موارد این محاسبهٔ تقریبی می‌تواند منجر به تغییر نتیجهٔ محاسبه، و یا تغییر عملیات‌های مورد نیاز برای محاسبه و یا حتی واگرایی محاسبه گردند.

آنچه مسلم است، یک عدد حقیقی نامتناهی داده را در خود جای داده است. در نتیجه یک ماشین تورینگ با توجه به توصیف متناهی‌اش نمی‌تواند محاسبه‌ای بر روی اعداد حقیقی را توصیف نماید. به عبارت دیگر برای بیان محاسبه‌ای بر روی اعداد حقیقی، نیاز به عملگرهایی (نامتناهی) بر روی اعداد حقیقی خواهیم داشت. البته این عملگرها می‌توانند عمل‌گرهایی بسیار ساده (همانند جمع و یا ضرب) باشند. از دیگر مشکلاتی که در مواجهه با اعداد حقیقی برای ماشین تورینگ ایجاد می‌شود حافظهٔ محدود ماشین تورینگ است. هر چند ماشین تورینگ طبق تعریف به دنباله‌ای نامتناهی از حروف دسترسی دارد، ولی در هر مرحلهٔ محاسبه می‌تواند دقیقا یک (و یا تعدادی متناهی) از خانه‌ها را مورد بررسی قرار دهدو در نتیجه محاسبات انجام شده توسط یک ماشین تورینگ نمی‌تواند پیچیدگی نامتناهی یک عدد حقیقی را در برگیرد.

۲.۱ مثال

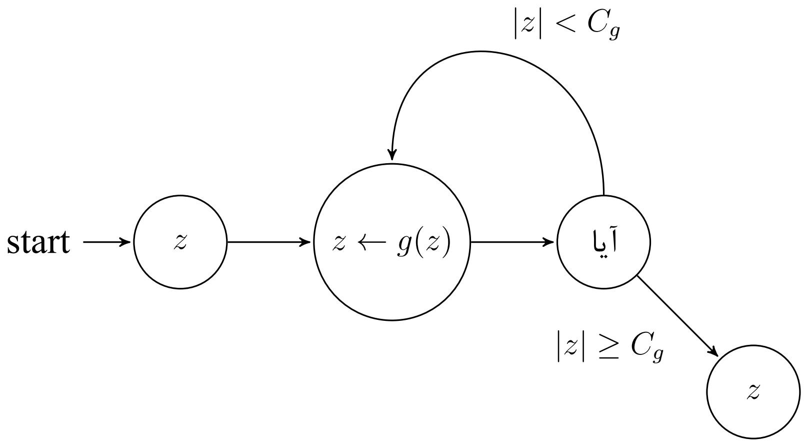
به عنوان مثال فرض کنید $g : \mathbb{C} \to \mathbb{C}$ نگاشتی چند جمله‌ای از $\mathbb{C}$ به روی خودش باشد. با توجه به اینکه g یک «درون‌ریخت»[۲] است می‌توان آن را به تعداد دلخواه تکرار کرد. به عبارت دیگر $g(g(z)) = g^{۲}(z)$ تعریف شده است و می‌توان برای تکرار kام g مقدار $g^k(z)$ را به ازای هر $z \in \mathbb{C}$ محاسبه کرد.

حال آیا با ورودی گرفتن عددی مانند z برای چند جمله‌ای مشخص g آیا مقدار $|g^k(z)|$ با افزایش مقدار k از هر مقدار عددی بزرگ‌تر خواهد شد، و یا در یک فضای کراندار باقی می‌ماند؟

لم ۱.۲. برای هر تابع چند جمله‌ای مانند g از درجهٔ ۲ یا بالاتر عددی حقیقی مانند C_g وجود دارد به طوری که اگر $|z| > C_g$ مقدار $|g^k(z)|$ با افزایش k به ∞ میل می‌کند.

اثبات: با توجه به اینکه g از مرتبه‌ای بالاتر از ۲ است برای مقادیر به اندازهٔ کافی بزرگ $|z|$ مقدار بزرگ‌ترین جملهٔ g عددی بسیار بزرگ خواهد شد و جملات دیگر نمی‌توانند آن را محدود نمایند. در نتیجه خواهیم داشت $|g(z)| > |z|$ و لم ثابت می‌شود. □

[۱] Scientific Computing
[۲] Endomorphism



شکل ۲.۱: ماشین بررسی واگرایی تابع g برای ورودی z

در ادامه ماشین $\lfloor m \rfloor$ را معرفی می‌نماییم که برای ورودی داده شده، به مسئلهٔ کران‌دار بودن تکرار g برای ورودی z پاسخ می‌دهد. نمودار این ماشین را می‌توانید در شکل ۲.۱ مشاهده کنید. با توجه به اینکه این ماشین در طول عملیات خود شرط $|z| < C_g$ بر روی فضای اعداد حقیقی را چک می‌کند، ماشین معرفی شده نیز بر روی اعداد حقیقی کار می‌کند.

می‌توان مشاهده کرد که مجموعهٔ کار $W_{\lfloor m \rfloor}$ دقیقا مجموعهٔ نقاطی است که در نتیجهٔ تکرار g به ∞ میل می‌کنند. $W_{\lfloor m \rfloor}$ در این ماشین مشابه یک مجموعهٔ شمارشی بازگشتی است.

در ادامه مجموعه‌ای از ماشین‌ها را تعریف می‌کنیم که نه تنها شامل ماشین $\lfloor m \rfloor$ باشد، بلکه ماشین‌هایی معادل ماشین‌های تورینگ را نیز در بر می‌گیرد. به $W_{\lfloor m \rfloor}$ یک مجموعهٔ R.E. بر روی حوزهٔ $\mathbb{R}$ می‌گوییم. توجه کنید که بر خلاف انتظار معمول از مجموعه‌های R.E.، $W_{\lfloor m \rfloor}$ شمارا نیست. سوال منطقی که می‌توان در این‌جا پرسید این است که آیا مجموعهٔ $W_{\lfloor m \rfloor}$ یک مجموعهٔ بازگشتی است؟ به عبارت دیگر آیا مکمل $W_{\lfloor m \rfloor}$ نیز یک مجموعهٔ R.E. است؟

۲.۲ محاسبه بر روی حلقهٔ دلخواه

بلام[۳]، شوب[۴] و اسمیل[۵] در سال ۱۹۸۹ در مقاله‌ای یک ماشین جهانی برای محاسبه بر روی یک حلقهٔ دلخواه مانند $\mathcal{R}$ معرفی کردند[۲]. این ماشین در حالت خاص $\mathcal{R} = \mathbb{Z}_2$ عملا توانی معادل ماشین تورینگ کلاسیک خواهد داشت. ساختار این ماشین نیز همانند ماشین تورینگ از مجموعه‌ای از حالات تشکیل شده که ماشین با انجام گذرهایی بر روی این مجموعه حالات به انجام عملیات محاسبه می‌پردازد. همچنین ماشین مجهز به مجموعه‌ای مرتب (و احتمالا نامتناهی) خانه‌های حافظه است که می‌تواند به دلخواه مقادیر آن‌ها را تغییر دهد و همانند نوار حافظهٔ یک ماشین تورینگ رفتار می‌کند. در مقابل ماشین معرفی شده

[۳] Lenore Blum
[۴] Mike Shub
[۵] Steve Smale

عملیات‌های جمع و ضرب تعریف شده بر روی حلقه را به عنوان عملیات‌های بدیهی در نظر می‌گیرد و می‌تواند در هر مرحله از محاسبه به تعداد متناهی و از پیش مشخصی از خانه‌های حافظهٔ خود دسترسی پیدا کند.

۲.۲.۱ تعریف

فرض کنید $\mathcal{R}$ یک «حلقهٔ یکدار جابجایی»[۶] باشد. همچنین فرض کنید $\mathcal{R}$ ترتیب داشته باشد. مثال‌های اصلی مورد بحث، مجموعهٔ اعداد صحیح، $\mathcal{R} = \mathbb{Z}$ و $\mathcal{R} = \mathbb{R}$، مجموعهٔ اعداد حقیقی است. همچنین $\mathcal{R}^n$ را حاصل ضرب دکارتی n بار $\mathcal{R}$ در خودش تعریف می‌کنیم. ماشین $[m]$ بر روی فضای $\mathcal{R}$ شامل فضای ورودی $\mathcal{I} = \mathcal{R}^i$، فضای خروجی $\mathcal{O} = \mathcal{R}^o$ و در نهایت فضای حالت $\mathcal{S} = \mathcal{R}^s$ است. ابتدا به بررسی حالتی می‌پردازیم که $i, o, s < \infty$ باشد. همچنین ماشین شامل یک گراف جهت‌دار است که هر راس می‌تواند یکی از چهار نوع زیر را داشته باشد:

- ماشین دقیقا یک راس از نوع ورودی دارد. این راس هیچ یال ورودی ندارد و دقیقا یک یال خروجی دارد. تابع متناظر با این گره نیز تابع خطی $I : \mathcal{I} \to \mathcal{S}$ است.

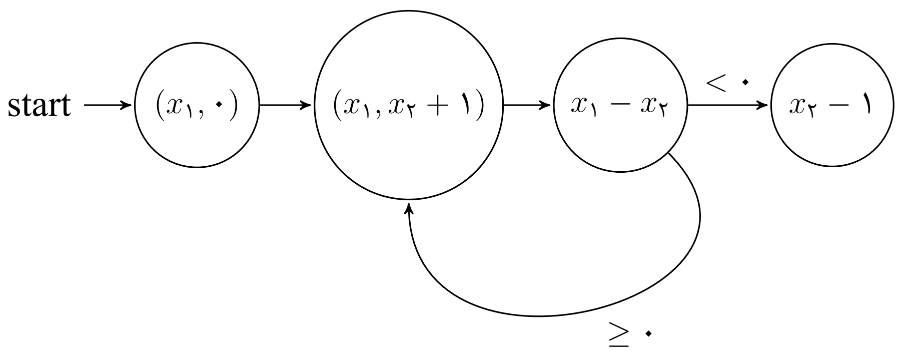
- ماشین دارای حداقل یک راس از نوع خروجی است. این راس هیچ یال خروجی‌ای ندارد و دقیقا یک یال ورودی دارد. همچنین تابع متناظر با این راس نیز تابعی خطی مانند $O_k : \mathcal{S} \to \mathcal{O}$ است.

- راس‌های محاسبه‌گر در ماشین دقیقا یک یال ورودی و یک یال خروجی دارند. متناظر با هر راس محاسبه‌گر نگاشت چندجمله‌ای $g_k : \mathcal{S} \to \mathcal{S}$ وجود دارد. در صورتی که حلقهٔ R یک «میدان»[۷] باشد، تابع g_k می‌تواند به صورت حاصل تقسیم دو چندجمله‌ای تعریف شود.

- راس‌های شرطی، که دقیقا یک ورودی و دو خروجی دارند. تابع متناظر با یک راس شرطی تصویری مانند $h_k : \mathcal{S} \to \mathbb{R}$ است. هر کدام از خروجی‌های راس شرطی متناظر با یکی از حالت‌های $h_k(x) < ۰$ و $h_k(x) \geq ۰$ هستند.

روش کار ماشین در ابتدای محاسبه در راس ورودی قرار دارد و در ادامهٔ محاسبه در هر مرحله با حرکت بر روی یال‌های گراف وارد راس‌های دیگری می‌شود. ماشین برای گذر از روی هر کدام از راس‌ها ابتدا تابع متناظر آن راس را با مقادیر داخل حافظه محاسبه و مقدار جدید را جایگزین مقدار قبلی می‌کند. سپس به راس بعدی بر روی گراف حرکت می‌کند. در حالتی که بر روی یک راس شرطی قرار داشته باشد، به جای جاگزین کردن مقدار تابع در حافظه، بسته به مقدار تابع یکی از دو مسیر موجود را ادامه می‌دهد. این روند تا زمانی که ماشین بر روی یک راس خروجی قرار بگیرد ادامه خواهد داشت و در نهایت حاصل عبارت راس خروجی را چاپ می‌کند.

[۶]Commutative Unitary Ring
[۷]Field



شکل ۲.۲: ماشین محاسبه‌گر $floor(x)$

مثال

فرض کنید با مدل معرفی شده بخواهیم تابعی را پیاده‌سازی کنیم که عددی مانند $x \geq 0$ را ورودی بگیرد و مقدار $floor(x)$ را محاسبه کند. گراف ۲.۲ معرف یک ماشین محاسبه‌گر برای این تابع است. ماشین با ورودی گرفتن x کار خود را آغاز می‌کند. این ماشین نیاز به دو خانهٔ حافظه برای انجام محاسبات خود دارد، در نتیجه در راس ورودی نگاشتی از فضای ورودی $\mathcal{I} = \mathbb{R}$ به فضای حالت خود که $\mathcal{S} = \mathbb{R}^2$ است قرار دارد. سپس در هر مرحله از حلقهٔ پردازش به مقدار x_2 یک واحد اضافه می‌کند. در نهایت زمانی که مقدار x_2 از مقدار x_1 بیشتر شد محاسبه پایان می‌یابد و مقدار $x_2 - 1$ به عنوان نتیجهٔ نهایی در خروجی چاپ می‌شود.

۲.۲.۲ محاسبه در فضای نامتناهی

آنچه تا کنون معرفی شده یک ماشین محاسبه‌گر بر روی حلقهٔ دلخواه $\mathcal{R}$ است، ولی خانوادهٔ تمام ماشین‌های تعریف شده در فضای متناهی نمی‌توانند یک ماشین جهانی داشته باشند. توجه کنید که اگر ماشینی تابع $f : \mathcal{R}^n \to \mathcal{R}^m$ را شبیه سازی کند، ماشین دیگری وجود خواهد داشت که تابعی مانند $g : \mathcal{R}^{n+1} \to \mathcal{R}^m$ را شبیه سازی می‌کند. در نتیجه اگر ماشین جهانی ورودی‌ای n بعدی داشته باشد، مجموعه‌ای از توابع را نمی‌تواند شبیه‌سازی کند.

برای رفع این مشکل اجازه می‌دهیم $n = \infty$ که در این صورت $\mathcal{R}^n$ مجموعه‌ای با شمارا نامتناهی بعد خواهد بود. افزایش ابعاد ورودی ماشین مشکلاتی را (از نظر محاسباتی) به دنبال خواهد داشت که با اعمال محدودیت‌هایی از بروز آن‌ها جلوگیری می‌کنیم.

مسئلهٔ اول پیچیدگی راس‌های محاسبه‌گر در تعریف ماشین است. با محدود کردن تابع چند جمله‌ای به مجموعه‌ای از توابع که تاثیری محدود (و محلی) بر روی مجموعهٔ داده دارند، می‌توانیم این پیچیدگی را کنترل کنیم و عبارت را به صورت موثری تبدیل به یک چند جمله‌ای معمولی با بعد متناهی کنیم. به عبارت دیگر در حالت خاص $n = \infty$ نگاشت f تنها زمانی چند جمله‌ای خواهد بود که عددی مانند k

وجود داشته باشد به طوری که به ازای هر $i > k$ داشته باشیم $f_i(x) = x_i$ و به ازای هر $i \leq k, j > k$ داشته باشیم $\partial f_i(x)/\partial x_j \equiv ۰$. به عبارت دیگر در محاسبهٔ تابع f تنها k مولفه فعال و تاثیر گذار هستند.

مسئلهٔ بعدی نگهداری یک مجموعهٔ $\mathcal{R}^\infty$ است. هر چند در نظریهٔ محاسبه همواره نوار حافظه طولی نامتناهی دارد، ولی در عمل طول نوار را بهتر است با عبارت «به قدر کافی بلند» جایگزین کنیم. این دو عبارت تفاوتی اساسی دارند: زمانی که از طول نامتناهی صحبت به عمل می‌آید، عملا نمی‌توان در دنیای واقعی تجسمی از نوار حافظه ارائه کرد. در مدل استاندارد ماشین تورینگ، هر چند نوار حافظه، طبق تعریف، طولی نامتناهی دارد، ولی در عمل فقط تعداد متناهی خانهٔ آن مقدار دهی شده‌اند و بقیهٔ خانه‌ها مقدار پیش‌فرض تهی را دارا هستند. می‌توان از همین شیوه برای معرفی یک مجموعهٔ $\mathcal{R}^\infty$ استفاده کرد. به بیان دقیق‌تر کافی‌ست زیر مجموعه‌ای از این فضا را در نظر بگیریم که به ازای اندیس‌های به قدر کافی بزرگ داشته باشیم $x_i = ۰$.

با توجه به تعریف تابع چند جمله‌ای $\mathcal{R}^\infty \to \mathcal{R}^\infty$ بر روی فضای نامتناهی ساختار یک ماشین محاسبه‌گر بر روی فضای $\mathcal{R}^\infty$ از بسیاری جهات مشابه یک ماشین با فضای حالت متناهی است. ولی رأس‌های تعریف شده در ساختار ماشین متناهی اجازهٔ دسترسی به خانه‌های با اندیس بسیار بزرگ در فضای حالت را به ما نمی‌دهند. در نتیجه ما نیاز به رأسی از نوع پنجم خواهیم داشت. این نوع از رئوس اجازه‌ی دسترسی به خانه‌های با اندیس بسیار بزرگ را برای ماشین فراهم می‌کند.

در ادامه فضای حالت ماشین را به $\bar{S} : \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathcal{R}^\infty$ گسترش می‌دهیم. یک عضو $\bar{S}$ ساختاری شبیه به $(i, j, x_۱, x_۲, \ldots)$ دارد. رأس نوع پنجم، همانند رأس‌های محاسباتی یک یال ورودی و یک یال خروجی دارد ولی بر خلاف دیگر انواع رئوس نگاشت متناظر با این رأس تابعی چندجمله‌ای نیست. این رأس در نتیجهٔ عملیات خود، مقدار خانهٔ iام را درون خانهٔ j ام می‌نویسد. به عبارت دیگر اگر تابع g_k نگاشتی متناظر با نوع پنجم باشد، به صورت زیر محاسبه می‌شود:

$$g_n(i, j, x_۱, x_۲, \ldots, x_i, \ldots, x_j, \ldots) = g_n(i, j, x_۱, x_۲, \ldots, x_i, \ldots, x_i, \ldots)$$

با توجه به تغییرات مجموعهٔ کار ماشین، بقیهٔ توابع نیز باید تغییراتی داشته باشند. تابع ورودی $I : \mathbb{R}^n \to \bar{S}$ را می‌شود به صورتی تعریف نمود که $I(x)_{۲k+۱} = x_k$ و همچنین مقادیر دو مؤلفهٔ صحیح را $i = ۱, j = ۱$ قرار دهد. به این ترتیب ماشین در میان داده‌های ورودی «فضای کافی» برای محاسبات خود خواهد داشت. همچنین به خاطر مسائل فنی فرض می‌کنیم که $I(x)_۴$ طول عبارت ورودی داده شده را مشخص کند. طول عبارت برای مقادیر x ناصفر برابر اندیس بزرگترین عنصر ناصفر ورودی است. همچنین اگر ورودی ماشین برابر ۰ بود، طول ورودی را مقدار ۱ قرار می‌دهیم.

در ادامه مقدار تابع محاسبه‌گر باید متناسب با تغییر فضای $\bar{S}$ تغییر کند. تعریف می‌کنیم تابع $g_k : \bar{S} \to \bar{S}$ می‌تواند نگاشت متناظر با یک رأس محاسباتی باشد هر گاه داشته باشیم $g_k(i, j, x) =$

$(i'(i,j), j'(i,j), x'(i,j,x))$ به طوری که $i + $ ۱ یا i یا $i'(i,j) =$ و ۱ یا ۱ $j + $ یا j یا $j'(i,j) =$. در نهایت تابع $x'(i,j,x) : \mathcal{R}^\infty \to \rangle\backslash\{\sqcup\dagger$ یک تابع چند جمله‌ای بر اساس تعریف تابع چند جمله‌ای بر روی یک فضای نامتناهی باشد.

در مورد رأس‌های شرطی نیز تصویر $h_k : \bar{S} \to \mathbb{R}$ باید تابعی چند جمله‌ای مطابق با تعریف گفته شده باشد. در نهایت برای معرفی دقیق‌تر ماشین $\lfloor m \rfloor$، عدد k_m را برابر اندیس بزرگترین درایه به کار رفته در گره‌های ماشین m معرفی می‌نماییم. به طور مشابه مقدار d_m نیز درجهٔ بزرگ‌ترین چند جمله‌ای در میان رأس‌های m است.

تابع متناظر با ماشین $\lfloor m \rfloor$ همانند حالت متناهی تعریف می‌گردد. اگر به ازای $n, l \leq \infty$ تابع $f : \mathcal{R}^n \to \mathcal{R}^l$ تعریف شده باشد می‌گوییم f محاسبه‌پذیر است، اگر و تنها اگر ماشینی مانند $\lfloor m_f \rfloor$ وجود داشته باشد به طوری که دامنهٔ f برابر دامنهٔ ϕ_{m_f} باشد و به ازای هر $x \in domain(f)$ داشته باشیم $\phi_{m_f}(x) = f(x)$. می‌گوییم دو ماشین مانند m و m' متشابه‌اند اگر و تنها اگر هر دو یک تابع را محاسبه نمایند.

مجموعه‌ای مانند $\mathcal{Y} \subset \mathcal{R}^n$ را مجموعهٔ شمارشی بازگشتی می‌نامیم هرگاه ماشینی مانند $\lfloor m \rfloor$ وجود داشته باشد به طوری که $\mathcal{Y} = W_m$. همچنین مجموعهٔ $\mathcal{Y}$ را تصمیم‌پذیر یا مجموعهٔ بازگشتی می‌نامیم هر گاه هم خود این مجموعه و هم متمم آن شمارشی بازگشتی باشند.

۲.۲.۳ ماشین استاندارد

برای راحت‌تر بررسی کردن ماشین‌های معرفی شده، گروهی از ماشین‌ها را به عنوان «ماشین استاندارد»[۸] معرفی می‌نماییم. یک ماشین حالت استاندارد دارد اگر داشته باشیم:

- به ازای هر رأس شرطی مانند رأس kام تابع شرط، قید $h_k(x) = x_1$ را ارضا کند.
- ماشین فقط یک رأس خروجی داشته باشد.
- رأس‌های ماشین با اعداد $1, 2, \ldots, N$ شماره گذاری شده باشند به طوری که
  - رأس شمارهٔ ۱ همواره رأس ورودی باشد.
  - رأس شمارهٔ N هموارهٔ رأس خروجی باشد.
- در حالتی که فضای ورودی متناهی است، تابع ورودی برابر نگاشت طبیعی از فضای ورودی به فضای کار ماشین باشد. همچنین چه در حالت متناهی و چه در حالت نامتناهی خروجی تابع نگاشتی طبیعی از فضای کار ماشین به فضای خروجی باشد.

[۸] Standard Machine

گزاره ۲.۲. به ازای هر ماشین دلخواه مانند $\lfloor m\rfloor$ بر روی حلقهٔ $\mathcal{R}$ یک ماشین استاندارد مشابه $\lfloor m\rfloor$ وجود دارد.

اثبات: شرط اول را می‌توان با اضافه کردن یک رأس محاسباتی قبل و بعد از هر رأس شرطی ارضا کرد. رأس قبل از رأس شرطی تمامی عناصر دنبالهٔ داده‌های ماشین را یک خانه به جلو حرکت می‌دهد و نتیجهٔ چندجمله‌ای درون رأس شرطی را در خانهٔ اول می‌نویسد. در گره‌هایی که بعد از رأس شرطی اضافه می‌شوند نیز تمام داده‌ها به مکان اولیهٔ خود باز می‌گردند.

برای شرط دوم کافی است تمام رأس‌های خروجی را تبدیل به رأس محاسباتی کنیم و همه را به یک رأس خروجی متصل نماییم.

با توجه به بخش سوم، وجود یک برچسب گذاری متناسب با شروط گفته شده واضح است.

برای بخش چهار نیز کافی‌ست یک رأس محاسباتی قبل از رأس خروجی و یک رأس محسباتی بعد از رأس ورودی اضافه نماییم.

□

۲.۳ ماشین جهانی

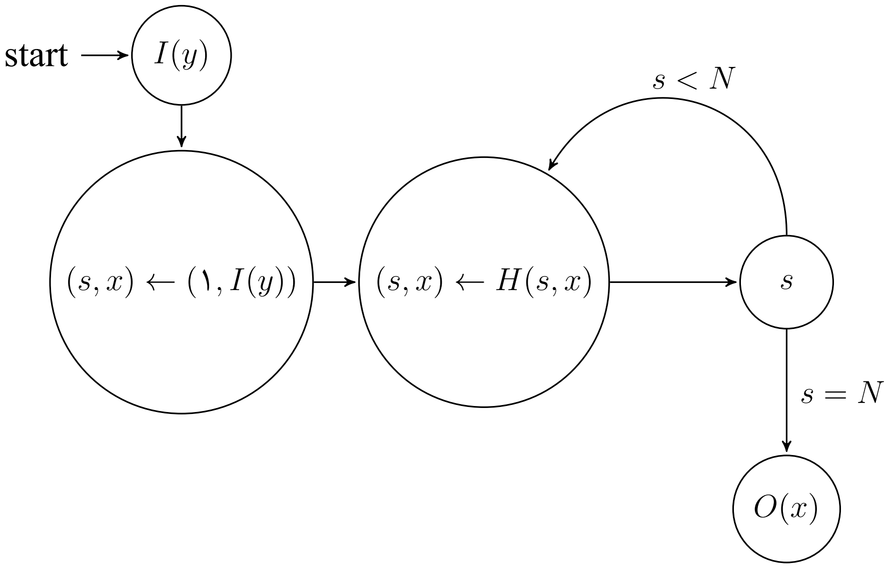
اگر ماشین $\lfloor m\rfloor$ یک ماشین استاندارد باشد، پیکربندی آن را می‌توان به صورت یکی از اعضای مجموعهٔ $\mathcal{N}\times\mathcal{S}$ نشان داد به طوری که $\mathcal{N}:\{۱,\ldots,N\}$ مجموعه‌ای از گره‌های ماشین $\lfloor m\rfloor$ است و $\mathcal{S}$ فضای حالت ماشین را مشخص می‌کند.

همچنین برای $\lfloor m\rfloor$ یک درون‌ریخت مانند $H_m:\mathcal{N}\times\mathcal{S}\to\mathcal{N}\times\mathcal{S}$ وجود دارد به طوری که $H(k,x)=(\beta(k,\chi(x)),g_k(x))$ به طوری که تابع β گرهٔ بعدی در محاسبه را مشخص می‌کند و $\chi(x):\mathcal{S}\to\mathbb{R}$ تابعی است که به صورت زیر تعریف شده است:

$$\chi(x)=\begin{cases} ۱ & x_۱\geq ۰\\ -۱ & x_۱<۰\end{cases}$$

در ادامه تابع $\beta:\mathcal{N}\times\{\pm ۱\}\to\mathcal{N}$ رابه صورت زیر تعریف می‌کنیم که تابعی وابسته به $\lfloor m\rfloor$ است.

$$\beta(k,\sigma)=\begin{cases} N & k=N\\ \beta(k) & \text{اگر } k<N \text{ و رأس } k \text{ شرطی نباشد}\\ \beta^{+}(k) & \text{اگر } k<N \text{ و رأس } k \text{ شرطی باشد و } \sigma=۱\\ \beta^{-}(k) & \text{اگر } k<N \text{ و رأس } k \text{ شرطی باشد و } \sigma=-۱\end{cases}$$



شکل ۲.۳: ساختار کلی یک ماشین جهانی

تابع g_n نیز در صورتیکه $k = ۱$ و یا $k = N$ و یا k یک راسی شرطی باشد، برابر تابع همانی است. در غیر این صورت اگر k گره‌ای محاسباتی و یا گرهٔ نوع پنجم باشد، $g_n(x)$ محاسبهٔ آن گره را انجام می‌دهد.

در نتیجهٔ آنچه گفته شد ماشین $[m]$ بر روی $\mathcal{R}$ با ورودی $y \in \mathcal{I}$ محاسبه‌ای را نمایش می‌دهد که به صورت $z_۰, z_۱, z_۲, \ldots, z_s, \ldots$ است به طوریکه $z_۰ = (۱, I(y))$ $z_s \in \mathcal{N} \times \mathcal{S}$. همچنین برای هر $s > ۰$ خواهیم داشت $H(z_{s-۱}) = z_s$. مطابق ادبیات نظریهٔ سیستم‌های دینامیکی، دنبالهٔ z_s یک مدار H است که از مبدا $z_۰$ شروع می‌شود. توجه کنید که اگر $[m]$ یک ماشین نامتناهی باشد و داشته باشیم $x_s = (i, j, \ldots)$ می‌توان نتیجه گرفت $i, j \leq s$.

با توجه به آنچه گفته شد می‌توان کارکرد یک ماشین بر روی فضای $\mathcal{R}$ را به صورت خلاصه شده مطابق نمودار ۲.۳ نمایش داد. نکتهٔ قابل توجه این است که تابع H خود یک تابع چند جمله‌ای است. به عبارت دیگر می‌توان معادل هر ماشین بر روی فضای $\mathcal{R}$ یک ماشین یافت که فقط شامل یک گرهٔ ورودی، یک گرهٔ محاسبه‌گر، یک گرهٔ شرطی و یک گرهٔ خروجی است. این ماشین متناظر کلید اصلی برای ساختن ماشین جهانی است.

ماشین جهانی کمابیش از نظر ساختاری همانند یک ماشین جهانی برای یک ماشین تورینگ ساده است. ماشین جهانی ابتدا با دریافت کردن رشتهٔ یک ماشین معادل ساده شدهٔ آن را محاسبه کند. سپس چرخهٔ ماشین شبیه‌سازی شده را تا زمانی ادامه می‌دهد که ماشین شبیه‌سازی شده متوقف شود و در نهایت جواب را چاپ می‌کند.

فصل ۳
ماشین‌های تعاملی

تعدادی از محققان مشاهده کرده‌اند که مدل محاسبهٔ ماشین تورینگ، که تمرکز بر نظریهٔ توابع محاسبه‌پذیر دارد، نمی‌تواند توصیف دقیق و جامعی از رایانه‌های امروزی، ارائه دهد. ون لوئن[۱] [۸] می‌گوید:

> ... الگوی کلاسیک تورینگ ممکن است دیگر توان در برگیری تمام جنبه‌های محسبات رایانه‌ای امروزی نداشته باشد.

همچنین میلنر[۲] در سخنرانی جایزهٔ تورینگ [۹] خود می‌گوید:

> در طول دههٔ هفتاد، من متقاعد شدم که تئوری پردازش موازی و تعامل نیاز به چهارچوبی جدید دارد. به این معنی که تنها بهبود آنچه ما به صورت طبیعی برای پردازش خطی از آن یاد می‌کنیم کافی نیست.

تعامل در ساده‌ترین شکل می‌تواند به صورت ارتباط دو طرفهٔ محیط و ماشین مدل شود. با این تفاوت که ماشین قبل از چاپ کردن خروجی به تمام دادهٔ ورودی دسترسی ندارد. حتی ممکن است بعضی از ورودی‌های ماشین متناسب با خروجی‌های آن تولید شوند. یک بازی شطرنج را اگر در نظر بگیرید، دو بازیکن مطابق آنچه تعریف شد با یکدیگر تعامل دارند. هر کدام از بازیکنان در نوبت خود حرکت طرف مقابل را به عنوان ورودی دریافت می‌کنند، و متناسب با آن حرکتی انجام می‌دهند.

همچنین وگنر [۱۰، ۱۱] نشان می‌دهد که مدل‌های تعاملی متناهی می‌توانند توصیف کامل‌تری نسبت به مدل‌های «الگوریتمی» که ماشین تورینگ توصیف می‌کند ارائه دهند. با توجه به آنچه بیان شد می‌توان این مسئله را بررسی نمود که چه مجموعه‌ای از افزونه‌ها را باید به ماشین تورینگ کلاسیک اضافه کرد که نتیجه بتواند خواص متمایز کنندهٔ تعامل را نیز در بر بگیرد. همچنین به عنوان یک هدف جانبی می‌خواهیم تغییرات ایجاد شده در مدل کلاسیک ماشین تورینگ به تعبیری حداقل میزان ممکن باشد.

۳.۱ مثال

یک ماشین جوابگوی خودکار تلفن را در نظر بگیرید. این ماشین را $[AM]$[۳] می‌نامیم. این ماشین باید بتواند پیغام را دریافت و آن را ذخیره کند، همچنین پیغام‌های ذخیره شده را پخش نماید و یا آن‌ها را پاک نماید. برای نمایش بهتر این منظور می‌توان ماشین را به صورت تابعی با دو ورودی و دو خروجی متناظر نمود. به این ترتیب که یکی از ورودی‌ها بیانگر ورودی داده‌شده به ماشین باشد و ورودی دیگر مقدار حافظهٔ ماشین را پیش از شروع به کار بر روی ورودی مشخص نماید. مقادیر خروجی هم بیانگر مقدار محاسبه شده توسط ماشین، و محتویات حافظه پس از اتمام کار ماشین خواهند بود. به این ترتیب ماشین AM متناظر

[۱] Jan van Leeuwen
[۲] Robin Milner
[۳] Answering machine

با تابعی به صورت زیر است.

$$\phi_{AM}(record\, Y, X) = (ok, XY)$$

$$\phi_{AM}(play, X) = (X, X)$$

$$\phi_{AM}(erase, X) = (done, \varepsilon)$$

طبق آنچه گفته شد این تابع دو ورودی دریافت می‌کند، که اولی دستور داده شده به ماشین، و دومی بیانگر حافظهٔ ماشین پیش از شروع محاسبه است. در مقابل خروجی این تابع یک زوج مرتب از کلمات است، که عبارت اول پاسخ ماشین (عبارت نوشته شده در نوار خروجی) پس از اتمام محاسبه است، و عبارت دوم نشانگر مقدار نوار حافظه در انتهای عملیات است.

روش کار ماشین $\lfloor AM \rfloor$ به این صورت است که اگر به آن ترتیب ورودی‌های

[record A, erase, record BC, record D, play, …]

داده شود، ماشین به ترتیب پاسخ‌هایی معادل

[ok, done, ok, ok, BCD, …]

خواهد داد. توجه کنید که دنبالهٔ ورودی‌ها و خروجی‌های ماشین می‌توانند دنباله‌ای نامتناهی باشند. در مجموع می‌توان گفت در نتیجهٔ ورودی‌های داده شده، تعامل زیر انجام شده، که هر زوج مرتب بیانگر یک بار محاسبه توسط ماشین است:

[(record A, ok), (erase, done), (record BC, ok), (record D, ok),(play, BCD), …]

همچنین توجه کنید برای معرفی یک تعامل، نیازی به بیان محتویات حافظهٔ ماشین نیست، چرا که در هنگام شروع به کار ماشین پیش از اولین محاسبه حافظهٔ ماشین خالی است و سپس، پس از هر مرحله از تعامل مقدار حافظهٔ ماشین به صورت یکتا مشخص می‌گردد.

۳.۲ ماشین تورینگ ماندگار

با توجه به آنچه گفته شد، مدل‌های مختلفی معرفی شده‌اند که بتوانند بخشی از کمبودهای ذکر شده را پوشش دهد. ماشین تورینگ ماندگار [۴] (PTM) یکی از این مدل‌هاست[۱]. یک ماشین تورینگ ماندگار از لحاظ ساختاری مشابه به یک ماشین غیر قطعی با سه نوار حافظه است. یک نوار برای خواندن ورودی، یک نوار برای انجام محاسبه که ماشین می‌تواند به دلخواه اطلاعات آن را بخواند و در آن اطلاعات بنویسد، و در نهایت نوار سوم که مخصوص نوشتن است. ماشین هنگامی که یک ورودی از محیط دریافت می‌کند، محاسبه را شروع می‌کند و در نهایت خروجی را از طریق نوشتن در نوار خروجی به محیط باز می‌گرداند. سپس این چرخه به صورت نامتناهی تکرار می‌شود. ماندگاری ماشین به این معنی‌ست که ماشین از حافظهٔ

[۴] Persistent Turing Machine

ماندگاری برخوردار است و آنچه در نوار حافظهٔ خود نوشته است از یک چرخهٔ محاسبه تا چرخهٔ بعد در حافظه باقی می‌ماند. هر چرخهٔ محاسبهٔ PTM معادل یک پردازش کامل ماشین تورینگ غیر قطعی است.

ماندگاری حافظه باعث می‌شود ورودی‌های دریافتی توسط ماشین بر محاسبهٔ آیندهٔ ماشین تاثیر بگذارد. به این ترتیب که زمانی که ماشین ورودی دریافت می‌کند می‌تواند به دلخواه از محاسبه‌ای که انجام داده است اطلاعاتی را در حافظهٔ خود نگه دارد، سپس در محاسبهٔ مربوط به ورودی‌های بعدی از حافظهٔ خود استفاده کند و جواب منحصر به فردی تولید نماید. این در حالی‌ست که اگر نوار حافظهٔ ماشین ماندگار نبود، این تاثیر از یک مرحلهٔ محاسبه به مرحلهٔ بعد امکان پذیر نبود و محاسبهٔ خروجی به ازای هر ورودی فقط و فقط به مقدار همان ورودی وابسته می‌بود. به این خصیصه در PTM «وابستگی زمانی»[۵] می‌گوییم.

۳.۲.۱ تعریف

همانطور که گفته شد، یک PTM از لحاظ ساختاری معادل یک ماشین تورینگ غیر قطعی با سه نوار حافظه (N3TM) است و تنها تفاوت در نحوهٔ تفسیر نوارهاست. ماشین در نوار دوم حافظهٔ خود که آن را نوار کاری نامیدیم می‌تواند اطلاعات را ذخیره کند. به عبارت دیگر مقدار این نوار از یک مرحله‌ی محاسبه تا محاسبهٔ بعدی تغییر نمی‌کند، و ماشین با هر گونه داده‌ای که در پایان محاسبهٔ یک عبارت در آن نوار قرار داده باشد، محاسبهٔ عبارت بعدی را شروع می‌کند.

گام خرد

در یک PTM «گام خرد»[۶] دقیقا معادل یک گام ماشین N3TM است.

گام کلان

فرض کنید $\lfloor M \rfloor$ یک PTM بر روی الفبای Σ باشد و عبارات w_i, w, w', w_o کلماتی از الفبای Σ باشند. می‌گوییم ماشین «گام کلان»[۷] انجام داده، $w \xrightarrow[\lfloor M \rfloor]{w_i/w_o} w'$، اگر و تنها اگر ماشین $\lfloor M \rfloor$ پس از پردازش عبارت w_i زمانی که بر روی نوارهای ورودی، کاری و خروجی آن به ترتیب عبارات w_i, w, ε نوشته شده باشد، پس از نوشتن w_o بر روی نوار خروجی توقف کند و عبارت w' بر روی نوار کاری آن نوشته شده باشد و در نهایت عبارت w_i بر روی نوار ورودی باقی بماند.

توجه کنید که محتوای نوار ورودی در انتهای یک گام کلان برابر مقدار آن در ابتدای همان گام است، که باعث می‌شود نوار ورودی صرفا به عنوان ورودی استفاده شود. همچنین نوار خروجی در ابتدای هر گام کلان مقداری تهی دارد که بر کاربرد آن تنها به عنوان خروجی تاکید دارد.

[۵] History-dependent
[۶] Micro-step
[۷] Macro-step

۳.۲.۲ ماشین جهانی

در صورتی یک PTM بتواند رفتار هر ماشین PTM دیگری را شبیه‌سازی کند به آن یک ماشین جهانی می‌گوییم. یک PTM جهانی مانند $\lfloor U \rfloor$ مشابه هر ماشین PTM دیگری یک N3TM است. این ماشین محاسبهٔ خود را با یک گام کلان آغاز می‌کند: در این مرحله عبارت مشخصهٔ ماشینی را که باید شبیه‌سازی شود را ورودی می‌گیرد. در گام‌های کلان بعدی ماشین ورودی‌های دریافت شده را به ماشین شبیه‌سازی شده می‌دهد و آن را شبیه‌سازی می‌کند. در طول این عملیات مقدار نوار کاری خود را نیز متناسب با محتوای نوار کاری ماشین شبیه‌سازی شده به روز رسانی می‌کند و خروجی‌های ماشین شبیه‌سازی شده را چاپ می‌کند. در ادامه تعریف دقیق‌تر ماشین جهانی را بیان می‌کنیم.

تعریف ۱.۳. فرض کنید $\lfloor U \rfloor$ یک PTM با الفبای Σ باشد. همچنین فرض کنید $M \in \Sigma^*$ رشتهٔ معرف یک PTM باشد به طوریکه $\lfloor M \rfloor$ نیز بر روی الفبای Σ کار کند. همچنین مقادیر w_i، w_o، w و w' رشته‌هایی از الفبای Σ باشند. در این صورت $\lfloor U \rfloor$ یک ماشین جهانی است اگر شرایط زیر برقرار باشند.

- $\lfloor U \rfloor$ یک گام کلان آماده‌سازی داشته باشد به صورتی که

$$\varepsilon \xrightarrow[\lfloor U \rfloor]{\langle M,w \rangle/\varepsilon} \langle M, w \rangle$$

- اگر ماشین $\lfloor M \rfloor$ با محاسبه‌ای مانند $w \xrightarrow[\lfloor M \rfloor]{w_i/w_o} w'$ متوقف شود، ماشین $\lfloor U \rfloor$ نیز پس از محاسبه‌ای مانند

$$\langle M, w \rangle \xrightarrow[\lfloor U \rfloor]{w_i/w_o} \langle M, w' \rangle$$

متوقف گردد.

- اگر ماشین $\lfloor M \rfloor$ پس از گرفتن ورودی w_i وارد حلقه‌ای نامتناهی شود، ماشین $\lfloor U \rfloor$ نیز پس از دریافت همان ورودی، نباید جوابی چاپ کند.

به عنوان مثال اگر یک PTM جهانی ماشینی مانند $\lfloor AM \rfloor$ که در بخش قبل معرفی شد را شبیه‌سازی کند، تعامل زیر درون ممکن است انجام شود.

[(AM, ε), (record A, ok), (erase, done), (record BC, ok),
(record D, ok),(play, BCD), …]

اثبات: فرض کنید $\lfloor m \rfloor$ یک PTM دلخواه باشد و $w \xrightarrow[\lfloor m \rfloor]{w_i/w_o} w'$ یک گام کلان دلخواه از مراحل محاسبهٔ این ماشین باشد. طبق تعریف $\lfloor m \rfloor$ یک ماشین تورینگ نیز هست، پس یک ماشین تورینگ با یک نوار حافظه مانند $\lfloor m' \rfloor$ با عملکرد مشابه $\lfloor m \rfloor$ وجود دارد. به عبارت دقیق‌تر ماشین m' با ورودی گرفتن $\langle w, w_i, \varepsilon \rangle$ خروجی $\langle w', w_i, w_o \rangle$ را چاپ می‌کند. حال فرض کنید که $\lfloor U \rfloor$ یک ماشین تورینگ جهانی

باشد. در نتیجه ماشین $\lfloor U \rfloor$ با گرفتن ورودی $\langle m', \langle w, w_i, \varepsilon \rangle\rangle$ رفتار ماشین $\lfloor m' \rfloor$ را شبیه‌سازی می‌کند و سپس مقدار $\langle w', w_i, w_o \rangle$ را خروجی می‌دهد.

حال فرض کنید ماشین $\lfloor U' \rfloor$ یک PTM باشد توانایی اجرا کردن $\lfloor U \rfloor$ را بر روی ورودی دلخواه داشته باشد. همچنین ماشین $\lfloor U' \rfloor$ در اولین گام کلان خود رشتهٔ معرف یک ماشین را ورودی بگیرد و آن را بر روی نوار خود ذخیره کند. در گام‌های کلان بعدی ماشین $\lfloor U' \rfloor$ می‌تواند ورودی داده شده را به همراه ورودی آن گام به ماشین $\lfloor U \rfloor$ تحویل دهد و خروجی $\lfloor U \rfloor$ را چاپ نماید. $\square$

۳.۳ ماشین‌های تعاملی با راهنما

هر چند ماشین‌های تورینگ ماندگار پاسخگوی توصیف تعامل میان ماشین و محیط خارج است، ولی پردازش در دنیای مدرن امروزی شرایط بسیار دیگری را نیز در بر دارد. به عنوان مثال شما ممکن است قطعاتی را به رایانهٔ خود اضافه کنید و یا قطعاتی را حذف نمایید. تغییر قطعات به معنای تغییر توانایی‌های ماشین است، و این تغییرات برای ماشین قابل پیش‌بینی نیست. همچنین یک رایانه، بیش از یک کانال ارتباطی با محیط اطراف دارد، یک رایانه ممکن از از طریق نمایشگر، بلندگو و شبکهٔ محلی اطلاعاتی را به محیط بیرون تحویل دهد، و در مقابل از طرف شبکهٔ محلی، صفحه کلید و موشواره[۸] اطلاعاتی را دریافت نماید. همچنین این درگاه‌های ارتباطی هر کدام روش کدگذاری مخصوص خود را دارند که البته همه برای رایانه شناخته شده است. برای پاسخگویی به این نیازهای ما ماشین‌های تعاملی با راهنما[۹] را معرفی می‌کنیم. [۳]

این ماشین‌ها از بسیاری جهات شبیه به ماشین‌های معمولی تورینگ هستند ولی سه خاصیت به آن‌ها اضافه شده است: راهنما، تعامل و عدم محدودیت در مدت عملیات.

۳.۳.۱ توابع راهنما

یک ITMA باید بتواند تغییرات احتمالی در سخت‌افزار و یا نرم‌افزار خود توسط کاربر را مدل کند. این تغییرات باید مستقل از ورودی (تا لحظهٔ انجام تغییر) باشند. در غیر این صورت ماشین می‌تواند با استفاده از تغییرات انجام شده پاسخ‌هایی از پیش آماده شده بدهد. همچنین با استدلالی مشابه، انتظار داریم که تغییرات انجام شده در ماشین گستردگی زیادی نداشته باشند. چرا که در صورتی که تغییرات بسیار بزرگ باشند، می‌توانند اطلاعاتی راجع به تمام سوالات که احتمال دارد از ماشین پرسیده شود را با خود انتقال دهند. برای ارضای این دو شرط، ما بر روی تغییرات احتمالی ماشین این شرط را اضافه می‌کنیم که تغییرات حتما باید تابعی تنها از زمان t باشند و توصیف این تغییرات حداکثر از مرتبهٔ چند جمله‌ای از t باشند.

برای شبیه‌سازی اطلاعات اضافی (و احتمالا محاسبه ناپذیر) که ماشین دریافت می‌کند ما از پیشگو استفاده می‌کنیم. به تعبیر دیگر یک پیشگو حکم تغییرات و بهبود سخت‌افزار ماشین را دارد و در نتیجه

[۸] mouse
[۹] Interactive Turing machine with advice

نسخه‌ای محدودتر از پیشگوها را استفاده می‌کنیم که مستقل از دادهٔ ورودی هستند. توابع «راهنما»[۱۰] برای این منظور نامزد مناسبی هستند. کارپ و لیپتون [۱۲] اولین بار برای بررسی پیچیدگی محاسبهٔ غیریکنواخت از این توابع بهره گرفتند.

تعریف ۲.۳. تابع راهنما تابعی مانند $f : \mathbb{N} \to \Sigma^*$ است. تابع راهنمای را محدود به $S(n)$ می‌نامیم هرگاه برای تمامی مقادیر n، طول $f(n)$ از مقدار $S(n)$ کمتر باشد.

در صورتی که یک ITMA ورودی‌ای از طول n داده شده باشد، ماشین فقط می‌تواند مقدار تابع راهنما را برای مقدار n بپرسد. برای بیان دقیق‌تر فرض کنید $\mathcal{C}$ کلاس از زبان‌ها (مسائل) باشد که توسط ماشین‌های تورینگ حل می‌شوند و $\mathcal{F}$ مجموعه‌ای از توابع راهنما باشد.

تعریف ۳.۳. کلاس $\mathcal{C}/\mathcal{F}$ شامل تمام زبان‌هایی مانند L است، به شرطی که زبانی مانند $L_1 \in \mathcal{C}$ و تابعی مانند $f \in \mathcal{F}$ وجود داشته باشند به طوری که $x \in L$ اگر و تنها اگر $\langle x, f(|x|)\rangle \in L_1$

مجموعهٔ $\mathcal{C}$ می‌تواند هر کدام از مجموعه‌های محاسبه‌پذیر مانند $\mathcal{P}$ (توابع محاسبه‌پذیر در زمان چند جمله‌ای)، $\mathcal{NP}$ (توابع محاسبه‌پذیر غیر قطعی در زمان چند جمله‌ای) ، $PSPACE$ (توابع محاسبه‌پذیر در حافظهٔ چند جمله‌ای) و غیره باشد. برای مجموعهٔ $\mathcal{F}$ نیز می‌توان کلاس توابع محدود به تابع لگاریتمی، و یا تابع چند جمله‌ای را مد نظر قرار داد.

۳.۳.۲ تعامل و پردازش نامتناهی

ماشین‌های تورینگ در مرحلهٔ بعد باید بتوانند از تعامل و پردازش نامتناهی پشتیبانی کنند. هیچ کدام از این خاصیت‌ها برای متخصصین علوم کامپیوتر جدید نیستند. به عنوان مثال در بررسی پردازش موازی و یا روش‌های ارتباطی و الگوریتم‌های توزیع شده تعامل از جملهٔ ابزارهای بنیادین است. محاسبات نامتناهی نیز در بررسی ω-automata تعریف شده و به خوبی بررسی شده است.

برای در برگیری خواص گفته شده، ماشین تورینگ باید علاوه بر راهنما مجهز به تعداد متناهی درگاه ورودی و یا خروجی باشد. درگاه‌های ورودی به ماشین اجازه می‌دهند که از محیط بیرون اطلاعات حرف به حرف دریافت کند. اگر در طول محاسبه اطلاعاتی برای ورودی دادن به ماشین وجود نداشت، درگاه‌های ورودی یک حرف ثابت خارج از الفبای Σ را تکرار می‌کنند. وضعیت‌های مشابه در مورد درگاه‌های خروجی نیز به وسیلهٔ تکرار همین حرف مشخص می‌گردد.

۳.۳.۳ عملکرد

یک ITMA در کلیت عملکرد خود شبیه به یک ماشین تورینگ معمولی است. ابتدا ماشین با نوار حافظهٔ خالی محاسبهٔ خود را آغاز می‌کند. در ادامه در هر مرحله از محاسبه، ماشین اطلاعاتی را که بر روی درگاه‌های

[۱۰] Advice

ورودی آمادهٔ استفاده هستند را می‌تواند بخواند. همزمان در صورتی که اطلاعاتی برای چاپ داشته باشد، می‌تواند این داده‌ها را بر روی هر کدام از درگاه‌های خروجی بنویسد. رفتار ماشین بر اساس مشخصات آن لحظه‌اش است: مقادیری که بر روی درگاه‌های ورودی آمادهٔ استفاده هستند، مقادیری که توسط اشاره‌گر ماشین مشخص شده‌اند و در نهایت وضعیت ماشین مجموعهٔ این مشخصات لحظه‌ای را تشکیل می‌دهند. در پاسخ به این مشخصات ماشین می‌تواند بر روی نوار حافظهٔ خود، و یا هر کدام از درگاه‌های خروجی اطلاعاتی چاپ کند. همچنین می‌تواند اشاره‌گر حافظه را یک خانه به چپ یا راست منتقل کند. اگر به ازای هر مجموعه از مشخصات احتمالی که ماشین با آن‌ها ممکن است رو به رو شود، رفتار ماشین تعریف شده باشد، ماشین برای مدت زمان نامتناهی به کار کردن ادامه خواهد داد. در طول پردازش ممکن است مقدار حافظهٔ استفاده شده از هر مقدار محدودی بزرگ‌تر شود. همچنین ماشین در هر لحظه از پردازش می‌تواند از راهنمای خود نیز استفاده کند. هر چند برای استفاده از راهنما در زمان t فقط اجازه دارد مقادیر t و یا کمتر را از راهنما بپرسد.

فصل ۴

یادگیری در ماشین‌های محاسبه‌گر

ماشین حساب علیرغم سرعت بسیار بالا محدودیت‌های بسیاری دارد. یک ماشین‌حساب می‌تواند جواب یک مسئلهٔ پیچیدهٔ ریاضی را به درستی (و احتمالا سریع‌تر از هر انسانی) محاسبه نماید، ولی تمام دانش یک ماشین‌حساب توسط یک برنامه‌نویس به آن داده شده است. همچنین ماشین‌حساب نمی‌تواند روش‌های حل جدیدی را فرا بگیرد و خارج از محدودهٔ تعریف شدهٔ کارش محاسبه‌ای را انجام دهد. به عبارت دیگر تنها دلیلی که یک ماشین‌حساب می‌تواند محاسبه‌ای را انجام دهد وجود برنامه‌نویسی است که توانایی حل مسئله را به ماشین‌حساب داده است.

از اولین روزهایی که صحبتی در مورد دستگاه‌های محاسبه‌گر به میان آمد، انسان همواره خیال ماشین‌هایی را در سر می‌پرورانده که خود توانایی یاد گرفتن داشته باشند و بتوانند محدودهٔ دانش و توانایی خود را بدون کمک یک برنامه‌نویس گسترش دهند. نمونه‌هایی از این گونه دستگاه‌ها را می‌توان در آثار رمان‌نویسان مشهوری همچون آسیموف[۱] دید. در طول زمان، ما موفق به ساخت ماشین‌هایی شدیم که هر کدام توانایی حل بسیاری مسائل را داشته‌اند، ماشین تورینگ، ماشین‌های تعاملی و ماشین محاسبهٔ بلام، همه نمونه‌هایی از ماشین‌حساب هستند. ساختار کلی همهٔ این ماشین‌ها یکسان است: مقداری داده به عنوان ورودی دریافت می‌کنند، بر روی داده‌ها محاسبه‌ای از پیش تعیین شده را انجام می‌دهند و نتیجهٔ محاسبات را خروجی می‌دهند. آن‌ها توانایی حل کردن بسیاری مسائل را دارا هستند، و این کار را با سرعتی بسیار بالا انجام می‌دهند، ولی همواره وجود برنامه‌نویس را برای تشریح روش مدل‌سازی و حل مسائل لازم دارند. آنچه می‌تواند یک ماشین را به دستگاهی فراتر از یک ماشین‌حساب ارتقا دهد، توانایی حل مسئله، بدون کمک یک برنامه‌نویس خارجی است. به عبارت دیگر «یادگیری ماشین»[۲] به دنبال حل مسائل مختلف نیست، بلکه به دنبال یافتن راه حلی برای مسئلهٔ «چگونگی یافتن راه حل» است.

با وجود پیچیدگی تعریف مسئلهٔ مورد بحث، در ۶ دههٔ اخیر تلاش‌های بسیاری برای مدل‌سازی یادگیری ماشین انجام شده است. از جملهٔ آن‌های می‌تواند به مدل‌های محاسبهٔ تکاملی، شبکه‌های عصبی، کلونی مورچه‌ها، یادگیری عمیق و ... اشاره کرد.

۴.۱ مثال

«بازشناسی الگو»[۳] یکی از مسائلی است که از دهه ۵۰ میلادی و حتی پیش از آن مورد بحث بوده است. از جمله نمونه‌های آن می‌توان به بازشناسی صوت، دست نوشته‌ها و تشخیص وجود اشیاء خاص در تصاویر اشاره کرد. یکی از دلایل توجه بسیار زیاد به این دسته از مسائل، کاربردهای عملی آن‌هاست. به عنوان مثال، ادارات پست برای تسریع فرآیند ارسال محموله‌های پستی، عملیات خواندن آدرس و کد پستی محموله را به رایانه‌ها واگذار کرده‌اند.

---

[۱] Isaac Asimov
[۲] Machine Learning
[۳] Pattern Recognition

اوهر[۴] و واسلر[۵] در مقاله‌ای [۱۳] در ابتدای دهۀ ۶۰ می‌نویسند:

> روش‌های بازشناسی الگو را می‌توان به دو نوع کلی تقسیم نمود: روش‌هایی که توسط یک برنامه‌نویس به طور خاص برنامه‌ریزی شده‌اند و روش‌هایی که می‌توانند در مواجهه با داده‌ها مجموعه‌ای از ضرایب و مقادیر خود را تغییر دهند و به عبارتی راه حل را فرا بگیرند. دستۀ اول نمی‌توانند بسیاری از داده‌ها را از یکدیگر تفکیک نمایند، داده‌هایی که به چشم انسان به صورت کاملا مشخصی تفاوت دارند، ولی بیان این تفاوت به زبان ماشین کاری بس دشوار است. از طرفی برنامه‌هایی که می‌توانند خود را تغییر دهند، هنوز، نتوانسته‌اند مسائلی را که جذاب هستند با دقت بالا حل نمایند.

به مرور زمان روش‌های مختلفی برای توصیف یادگیری توسط ماشین ارائه شدند، و الگوریتم‌های آموزش پذیر توانستند پاسخ‌های بهتری به مسائل بدهند. امروزه روش‌هایی برای بازشناسی اعداد دست‌نویس ابداع شده‌اند که با خطایی کمتر از ٪۰٫۲ می‌توانند تصاویر را به درستی طبقه بندی نمایند[۱۴].

از میان روش‌هایی که خود توانایی یادگیری دارند، به طور معمول روش‌ها به دو دستۀ کلی «یادگیری با نظارت»[۶] و «یادگیری‌بدون نظارت»[۷] تقسیم می‌شوند. در یادگیری با نظارت، مجموعه‌ای از داده‌ها به همراه خروجی‌های مورد انتظار به الگوریتم یادگیری ارائه می‌شود و از ماشین انتظار می‌رود با استفاده از داده‌های یادگیری بتواند ارتباطی منطقی بین ورودی و «برچسب»[۸] پیدا کند. در مقابل روش‌های یادگیری بدون نظارت باید بتوانند بدون کمک خارجی بین داده‌ها ارتباط منطقی پیدا کنند و آن‌ها را طبقه‌بندی کنند. در نهایت روش‌های «یادگیری نیمه‌نظارتی»[۹] از هر دو نوع روش قبل در فرآیند آموزش استفاده می‌کنند و داده‌های آموزشی آن‌ها شامل هر دو گروه داده‌های برچسب‌دار و بدون برچسب است. در ادامه مثال از هر کدام از روش‌های با نظارت و بدون نظارت را مطرح می‌کنیم و آن‌ها را مورد بررسی قرار می‌دهیم.

۴.۲ شبکه‌های عصبی

از جملۀ روش‌های یادگیری با نظارت می‌توان به «شبکۀ عصبی»[۱۰] اشاره کرد. هر چند ایدۀ اولیه طراحی شبکه‌های عصبی به دهۀ ۵۰ میلادی باز می‌گردد[۱۵]، ولی یکی از اساسی‌ترین پیشرفت‌های در بررسی و استفاده از شبکه‌های عصبی با معرفی روش‌های «پس انتشاری»[۱۱] در ۱۹۷۵ اتفاق [۱۶] افتاد. از جملۀ

[۴] Leonard Uhr
[۵] Charles Vossler
[۶] Supervised Learning
[۷] Unsupervised Learning
[۸] Label
[۹] Semisupervised Learning
[۱۰] Neural Network
[۱۱] Backpropagation

کاربردهای این روش‌ها می‌توان به بازشناسی دست‌نوشته[۱۷]، صوت[۱۸]، داده‌کاوی[۱۹]، پیش‌بینی بازار بورس[۲۰] و بسیاری مسائل دیگر اشاره کرد.

(RNN)«شبکه‌های عصبی بازگردنده»[۱۲] به خاطر توانایی انجام تبدیل‌های بسیار پیچیده بر روی انواع داده‌ها از دیگر روش‌های یادگیری ماشین متمایز شده‌اند. همچنین ثابت شده است که یک RNN توانایی انجام پردازش دلخواه بر روی انواع داده را دارد و به عبارت دیگر «تورینگ کامل»[۱۳] است [۲۱]. ولی آنچه به صورت تئوری اثبات می‌شود همواره در عمل به سادگی انجام نمی‌شود، در نتیجه گریوز[۱۴]، وین[۱۵] و دنیهلکا[۱۶] در مقاله‌ای[۶] اقدام به معرفی ماشینی بر مبنای RNN کردند. تفاوت اصلی «ماشین تورینگ عصبی»[۱۷] با ماشین‌های تورینگ معمولی در مشتق‌پذیر بودن رفتار این ماشین‌هاست؛ در نتیجه می‌توان آن‌ها را با استفاده از الگوریتم‌های «یشترین کاهش»[۱۸] آموزش داد که منجر به روشی عملیاتی برای آموزش روش اجرا (در مقابل تاثیر مستقیم برنامه‌نویس) می‌شود.

یکی از کلیدی‌ترین تحولات در RNN معرفی LSTM[۱۹] بود[۲۲]. این ساختار کلی برای یک مشخص مشخص ابداع شد، برای رفع مشکل «محو شدن یا از انفجار شیب». LSTM این مشکل را معرفی یک انتگرال‌گیر کامل[۲۰] در ساختار حافظهٔ ماشین رفع می‌کند. این انتگرال‌گیر می‌تواند فرم ساده‌ای مانند $x_{t+۱} = x_t + i_t$ داشته باشد، که i_t ورودی ماشین در زمان t است. در صورتی که ما ساختاری مانند $g(\text{context})$ به این انتگرال‌گیر اضافه کنیم می‌توانیم کنترل کنیم که ماشین چه زمانی به ورودی‌ها توجه می‌کند و چه زمانی ورودی‌ها را بدون توجه رد می‌کند. به عبارت دقیق‌تر انتگرال‌گیر ما صورتی معادل $x_{i+۱} = x_t + g(\text{context})i_t$ پیدا خواهد کرد.

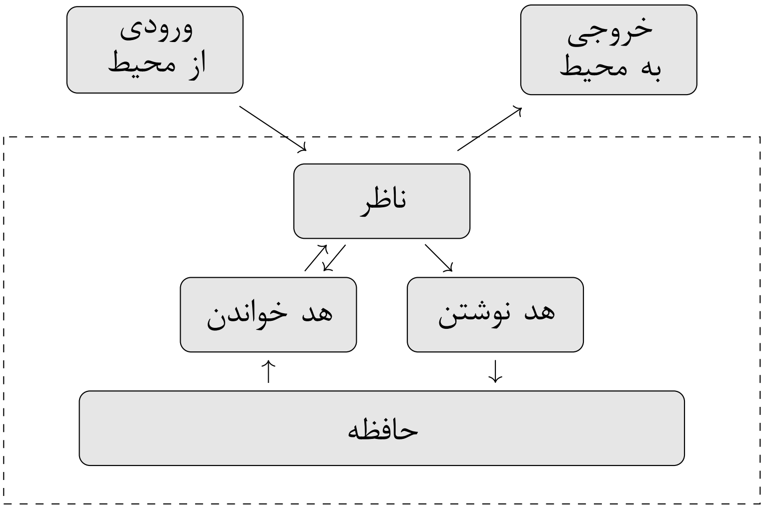
۴.۳ ماشین تورینگ عصبی

۴.۳.۱ تعریف

یک NTM همانند یک ماشین تورینگ از یک نوار حافظه تشکیل شده است و مجموعه‌ای از حالات وظیفهٔ تنظیم رفتار ماشین را به عهده دارند، ولی شباهت این دو ماشین در این‌جا به پایان می‌رسد. حالت ماشین بر خلاف ماشین تورینگ ساده که توسط مجموعه‌ای از حالات کنترل می‌گردد، در یک NTM کنترل ماشین بر عهدهٔ یک شبکهٔ عصبی است.

ماتریس M_t نوار حافظه یک NTM را در زمان t تشکیل می‌دهد که ماتریسی $N \times M$ بعدی از اعداد

[۱۲] Recurrent neural network
[۱۳] Turing Complete
[۱۴] Alex Graves
[۱۵] Greg Wayne
[۱۶] Ivo Danihelka
[۱۷] Neural Turing Machine
[۱۸] Gradient Descent
[۱۹] Long short-term memory
[۲۰] Perfect integrator



شکل ۴.۱: ساختار کلی یک NTM. مستطیل خط‌چین حد فاصل ماشین با محیط بیرون را مشخص می‌کند.

حقیقی است. در این ماتریس N نشانگر تعداد خانه‌های حافظه است و M برابر بعد هر خانه است. توجه کنید که مقادیر M و N جزو خواص ماشین هستند و در طول محاسبه تغییر نمی‌کنند.

خواندن و نوشتن در یک NTM توسط اشاره‌گر خواندن و نوشتن انجام می‌گردد. طبق تعریف ماشین می‌تواند بیش از یک اشاره‌گر برای خواندن و یا نوشتن داده داشته باشد. همچنین هر اشاره‌گر ماشین NTM می‌تواند در یک مرحله از عملیات چندین خانهٔ حافظه را همزان بخواند و به عبارتی توجه خود را میان مجموعه‌ای از خانه‌ها تقسیم نماید. شکل ۴.۱ ظاهری کلی از یک ماشین را نشان می‌دهد.

خواندن

برای هر کدام از اشاره‌گرهای خواندن مقدار w_t مستقلا برداری N بعدی است که در زمان t توسط ناظر محاسبه اعلام می‌گردد. w_t برداری نرمال شده است، به این معنی که برای مؤلفه‌های آن مانند $w_t(i)$ داریم:

$$\sum_i w_t(i) = ۱, \qquad ۰ \leq w_t(i) \leq ۱, \quad \forall i$$

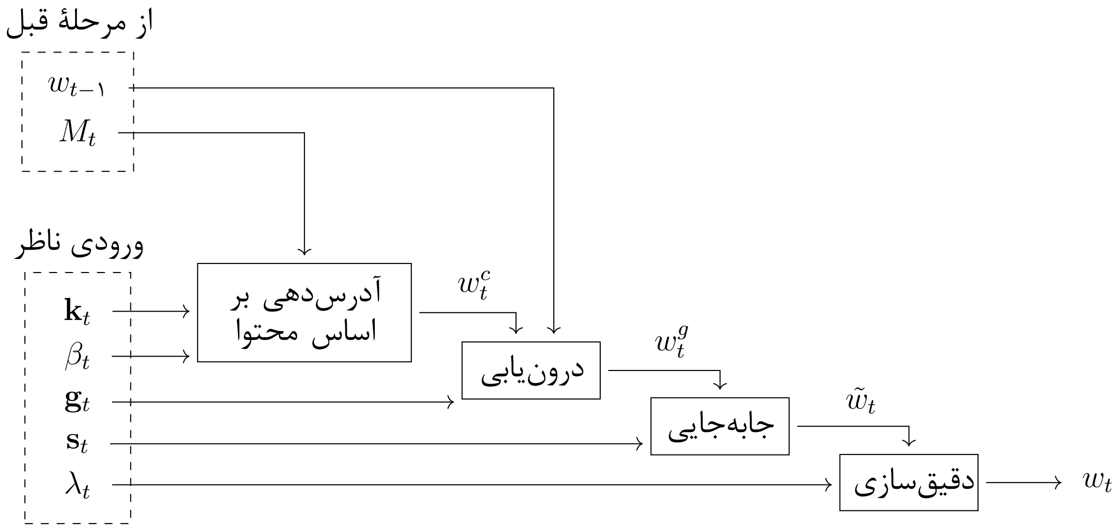
در عملیات خواندن، ماشین کل خانه‌های ماتریس را می‌خواند و ترکیب محدب این بردارها را با «توجه»[۲۱] به بردار وزن داده شده باز می‌گرداند. در نهایت بردار M بعدی r_t خوانده می‌شود.

$$r_t = \sum_i w_t(i)\mathbf{M}_t(i)$$

نوشتن

عملیات نوشتن از دو بخش حذف و تصحیح داده تشکیل شده است.

[۲۱] Attention



شکل ۴.۲: نموار ساختار آدرس‌دهی

در هر مرحله از پردازش ابتدا بردارهای w_t توسط ناظر محاسبه (و احتمالا متفاوت از بردارهای خواندن) برای هر کدام از اشاره‌گرهای نوشتن مشخص می‌گردد که میزان توجه ناظر به هر کدام از خانه‌های حافظه را مشخص می‌کند و برداری N بعدی است. بردار w_t همانند بردار متناظرش در مرحلهٔ خواندن برداری نرمال شده است. سپس بردار e_t مشخص می‌گردد که برداری M بعدی است و مشخص می‌کند هر کدام از عناصر یک خانهٔ حافظه به چه نسبتی باید پاک گردند. برای بردار e_t داریم

$$۰ \leq e_t(i) \leq ۱ \; \forall i$$

در نهایت بردار a_t مشخص می‌گردد، این بردار نیز برداری M بعدی است و میزان اضافه شدن داده به هر مولفه از خانه‌های حافظه را مشخص می‌کند. با معلوم شدن هر سه بردار مقادیر ماتریس $\mathbf{M}_t$ به صورت زیر به روز رسانی می‌گردد.

$$\mathbf{M_{t+۱}}(i) \leftarrow \mathbf{M_t}(i)(۱ - w_t(i)e_t) + w_t(i)a_t, \forall ۱ \leq i \leq N$$

آدرس‌دهی حافظه

هر چند روش کلی خواندن و نوشتن در حافظه را توصیف کرده‌ایم، ولی نحوهٔ وزن‌دهی بردارهای خواندن و نوشتن را هنوز توضیح نداده‌ایم. این بردارها نتیجهٔ ترکیب دو شیوهٔ مختلف آدرس‌دهی هستند. در شیوهٔ اول که به آن نام «آدرس‌دهی بر اساس محتوا»[۲۲] می‌دهیم، توجه ناظر بر اساس شباهت بین خانه‌های حافظه و یک بردار داده که توسط ناظر تعیین شده، مشخص می‌گردد. عملکرد این شیوه همانند عملکرد آدرس‌دهی

[۲۲] Content-based Addressing

بر اساس حافظه است که در شبکه‌های هاپ‌فیلد[۲۳][۲۳] مورد استفاده قرار گرفته است. مزیت این شیوه در آسانی بازیابی داده است: کافی است ناظر بتواند تقریبی از دادهٔ مورد نیاز خود را تولید کند، و با استفاده از این سیستم می‌تواند مقدار دقیق را به آسانی بیابد.

از طرف دیگر این شیوهٔ بازیابی اطلاعات همواره کارآمد نیست. در بعضی از مسائل ما نیاز داریم که مجموعه‌ای از داده‌ها را صرفا بر اساس جایگاه‌شان (بدون توجه به مقادیر آن‌ها) مورد بررسی قرار دهیم. مسائل محاسباتی از این دسته هستند. به عنوان مثال، برای محاسبهٔ تابع $f(x,y) = x \times y$ مقادیر x و y تاثیری در نحوهٔ محاسبهٔ ما ندارند، و آن‌چه اهمیت دارد صرفا جایگاه این دو عدد است. ناظر باید بتواند مقادیر x و y را ورودی بگیرد و آن‌ها را در حافظه ذخیره کند، در ادامهٔ کار این مقادیر را دوباره از حافظه بازیابی کند و عملیات ضرب را انجام دهد. ولی این روش آدرس‌دهی صرفا بر اساس جایگاه داده‌هاست و مقادیر آن‌ها تاثیری بر توجه ماشین نباید داشته باشد. در ادامه به این شیوه «آدرس‌دهی بر اساس موقعیت»[۲۴] خواهیم گفت. آدرس‌دهی بر اساس محتوا اکیدا جامع‌تر از آدرس‌دهی بر اساس موقعیت است، چرا که محتوای یک خانهٔ حافظه می‌تواند شامل آدرس آن خانه نیز باشد. ولی مطابق با آزمایش‌هایی که انجام دادیم، برای انجام بعضی از محاسبات و کلی‌تر کردن توانایی‌های ماشین، آدرس‌دهی بر اساس موقعیت جزو پیش‌نیازهای اساسی ماشین است. نمودار ۴.۲ ساختار کلی آدرس‌دهی را نشان می‌دهد.

دهی بر اساس محتوا برای آدرس‌دهی بر اساس محتوا، هر کدام از اشاره‌گرها ابتدا برداری M بعدی به عنوان کلید $\mathbf{k}_t$ تولید می‌کند. سپس این کلید با متر $K[.,.]$ با تک تک خانه‌های حافظه مقایسه می‌شود. در نتیجهٔ این عملیات بردار نرمال شدهٔ w_t^c بر اساس شباهت مقادیر درون حافظه و کلید داده‌شده محاسبه می‌گردد. همچنین مقدار β_t برای تضعیف و یا تقویت دقت توجه به کار می‌رود. در نهایت رابطهٔ زیر را خواهیم داشت:

$$w_t^c(i) = \frac{exp\left(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(i)]\right)}{\sum_j exp\left(\beta_t K[\mathbf{k}_t, \mathbf{M}_t(j)]\right)}$$

دهی بر اساس موقعیت آدرس‌دهی بر اساس محتوا، مکانیزمی است که حرکت‌های متوالی بر روی جدول حافظه را آسان‌تر می‌کند هم اجازهٔ دسترسی تصادفی به حافظه می‌دهد. این عملیات به وسیلهٔ یک «جابجایی چرخشی»[۲۵] بر روی وزن‌هاست. به عنوان مثال اگر تمام توجه یک اشاره‌گر بر روی یک خانه از حافظه باشد، پس از جابه‌جایی ۱ واحدی، تمام توجه ماشین بر روی خانهٔ بعدی حافظه خواهد بود. همچنین اگر خانهٔ مورد توجه آخرین خانهٔ جدول حافظه باشد پس از یک واحد جابه‌جایی چرخشی اولین خانهٔ جدول حافظه مورد توجه قرار خواهد گرفت. اگر مقدار جابجایی منفی باشد نیز خانه‌های قبلی مورد توجه ماشین

[۲۳] Hopfield networks
[۲۴] Location-based Addressing
[۲۵] Convolutional Shift

قرار خواهند گرفت و چرخش در خلاف جهت انجام می‌شود.

قبل از شروع چرخش، هر اشاره‌گر در مرحلهٔ درونیابی توجه، عدد g_t را از بازهٔ $[۰,۱]$ مشخص می‌کند. این عدد مشخص می‌کند که توجه اشاره‌گر چقدر متأثر از بردار توجه در گام پیشین باشد، و چقدر از مقادیر w_t^c تاثیر بپذیرد. نتیجهٔ این مقادیر بردار w_t^g خواهد بود.

$$w_t^g(i) = g_t w_t^c + (۱ - g_t) w_{t-۱}$$

اگر g_t مقدار صفر را اختیار کند، آدرس‌دهی بر اساس محتوا به طور کلی نادیده گرفته می‌شود، و در حالت حدی مقابل، اگر مقدار یک داشته باشد، بردار توجه در مرحلهٔ قبل نادیده گرفته می‌شود.

پس از مرحلهٔ درونیابی، هر اشاره‌گر مقدار جابه‌جایی را مشخص می‌کند. این مقدار به صورتی عددی حقیقی است. در ادامه بردار s_t محاسبه می‌شود. این عدد به کران پایین و بالای یک جابه‌جایی با عرض یک بر روی بردار وزن‌ها را مشخص می‌کند. به عنوان اگر مثال مقدار جابه‌جایی عدد ۷,۶ باشد، مؤلفهٔ $\mathrm{s}_t(۶) = ۰,۳$ و $\mathrm{s}_t(۷) = ۰,۷$ خواهد بود. در مثال فوق بقیهٔ عناصر دنبالهٔ s_t برابر ۰ خواهند بود. در ادامه با توجه بردار جابه‌جایی مقادیر $\bar{w}_t$ محاسبه می‌گردند.

$$\tilde{w}_t(i) = \sum_{j=۰}^{N-۱} w_t^g(j) s_t(i-j)$$

در رابطهٔ بالا تمام محاسبات صحیح بر باقیماندهٔ N انجام شده اند. حاصل معادلهٔ فوق در طول پردازش های متوالی ممکن است منجر به غیر دقیق شدن توجه شوند. به عنوان مثلا اگر توجه در مرحله‌ای متمرکز باشد، و در دو مرحلهٔ متوالی ابتدا عدد جابه‌جایی مقدار ۰,۵ و سپس مقدار ۰,۵− داشته باشد، بر خلاف انتظار نتیجه تمرکز زیادی نخواد داشت. برای تصحیح این پدیده در آخرین مرحلهٔ آدرس‌دهی هر اشاره‌گر عددی مانند $\lambda_t \geq ۱$ را مشخص می‌کند. سپس با توجه به رابطهٔ زیر بردار توجه «متمرکز»[۲۶] می‌شود و مقادیر نهایی این بردار محاسبه می‌گردند.

$$w_t(i) = \frac{\tilde{w}_t(i)^{\lambda_t}}{\sum_j \tilde{w}_t(j)^{\lambda_t}}$$

مجموعهٔ این سیستم آدرس‌دهی این توانایی را به ماشین می‌دهد که بتواند صرفا بر اساس محتوا آدرس دهی کند، و یا صرفا بر اساس بردار توجه قبلی و تغییرات مورد نیاز عملیات آدرس‌دهی را انجام دهد، و یا در همزمان ترکیبی از این دو را استفاده کند.

ناظر

ماشین تورینگ عصبی که معرفی شد تعداد بسیار زیادی پارامتر آزاد برای تنظیم و بهینه‌سازی عملکرد دارد. از جملهٔ این پارامترها می‌توان به تعداد اشاره‌گرهای خواندن و نوشتن، ابعاد ماترس حافظه و حداکثر میزان

[۲۶] Sharpen

جابجایی اشاره کرد. همچنین یکی دیگر از پارامترهای اساسی که ساختار ماشین را مشخص می‌کند، انتخاب شبکهٔ عصبی کنترل‌کنندهٔ ماشین است. این شبکه می‌تواند یک RNN باشد و یا رفتار ماشین توسط یک «شبکهٔ عصبی پیش‌خوراند»[۲۷] کنترل شود. استفاده از RNN این مزیت را دارد که واحد نظارتی ماشین از یک حافظهٔ داخلی (مستقل از ماتریس حافظه) بهره‌مند خواهد شد. به عنوان مثال اگر یک ماشین عصبی را معادل یک رایانهٔ امروزی در نظر بگیریم، واحد نظارتی ماشین تقریبا معادل پردازندهٔ مرکزی خواهد بود و ماتریس حافظه نقشی معادل حافظهٔ تصادفی[۲۸] را خواهد داشت. در این تشابه فعال‌سازی‌های پنهان RNN تقریبا معادل با ثبات[۲۹] را در پردازنده‌های امروزی خواهند داشت. از طرف دیگر یک شبکهٔ پیش‌خوراند می‌تواند همین رفتار را با تکرار نوشتن و خواندن مقادیر در یک خانهٔ مشخص از حافظه تقلید کند. از طرف دیگر تحلیل و بررسی رفتار یک شبکهٔ پیش‌خوراند به مراتب راحت‌تر از یک RNN است.

۴.۳.۲ بررسی توان محاسباتی

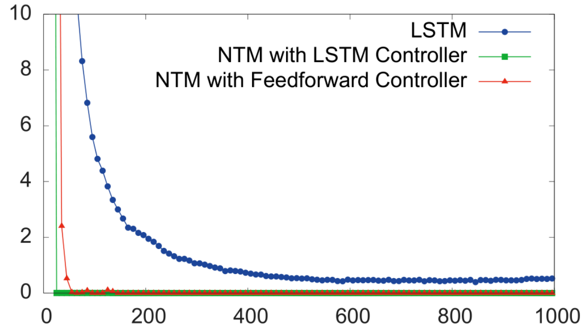
توانایی محاسبهٔ یک NTM همانند هر ساختار ریاضی دیگر قابل موشکافی است. این بررسی‌ها می‌تواند صرفا جنبهٔ عملیاتی و آزمایشگاهی داشته باشند، و یا بر پایهٔ استدلال ریاضی بنا شده باشند. به وضوح اثباتی که مبنای محکم منطقی داشته باشد بسیار قدرتمندتر است، ولی این گونه اثبات‌ها همواره از پیچیدگی بالایی برخوردار بوده‌اند. به وضوح ساختمان NTM به قدری پیچیده است که کار را برای یک تحلیل دقیق بسیار دشوار می‌سازد در نتیجه در ادامه به ذکر چند مثال از توانایی‌های این ماشین در عمل اکتفا می‌کنیم.

در این بخش به بعضی از الگوریتم‌های ساده مانند رونگاری داده و یا مرتب‌سازی آن اشاره می‌کنیم. منظور از این بخش صرفا توجیه کارآمدی یک ماشین تورینگ عصبی نیست، بلکه می‌خواهیم نشان دهیم که این ماشین‌ها می‌توانند الگوریتم‌هایی که بعضا بیش از فقط تحلیل محاسباتی بر روی داده‌ها هستند را هم آموزش ببینند و انجام دهند. چنین ماشین‌هایی زمانی کارآمد خواهند بود که بتوانند فراتر از آنچه آموزش گرفته‌اند محاسبات را انجام دهند. به عنوان مثال ماشینی که رونگاری کردن یک دنبالهٔ ۲۰ خانه‌ای را آموزش دیده است بتواند یک دنبالهٔ ۱۰۰ خانه‌ای را نیز به درستی رونگاری کند.

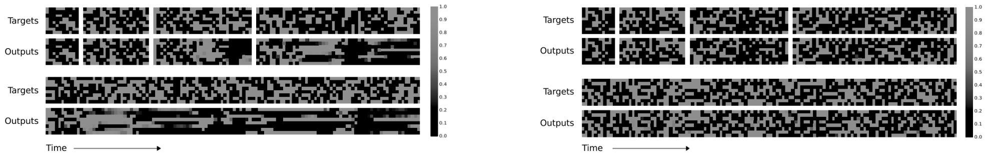
همچنین با توجه به پیچیدگی تحلیل محاسباتی این ماشین‌ها صرفا به تحلیل عملیاتی آن‌ها می‌پردازیم. در نهایت لازم به ذکر است که تمامی آزمایش‌ها بر روی ۳ مدل تکرار شده‌اند: ماشین عصبی با ناظر RNN، ماشین عصبی با ناظر شبکه‌ی عصبی پیش‌خوراند و در نهایت یک شبکهٔ LSTM.

در هر مرحله از محاسبه ماشین، ناظر می‌تواند از محیط بیرون ورودی بخواند و یا به محیط خروجی پاسخ دهد. همچنین ناظر مشخص می‌کند که کدام بخش از حافظه مورد بررسی قرار بگیرد و اطلاعات به چه ترتیبی خوانده و یا نوشته شوند.

[۲۷] Feedforward Neural network
[۲۸] Random-access memory
[۲۹] Register



شکل ۴.۳: نرخ همگرایی NTM و LSTM در مسئلهٔ رونگاری



(آ) NTM (ب) LSTM

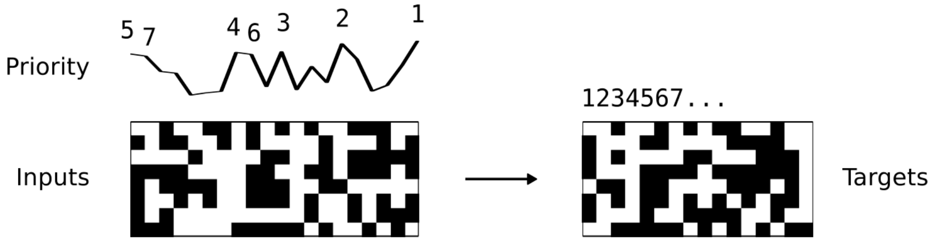
شکل ۴.۴: مقایسهٔ ورودی و خروجی در مسئلهٔ رونگاری

رونگاری

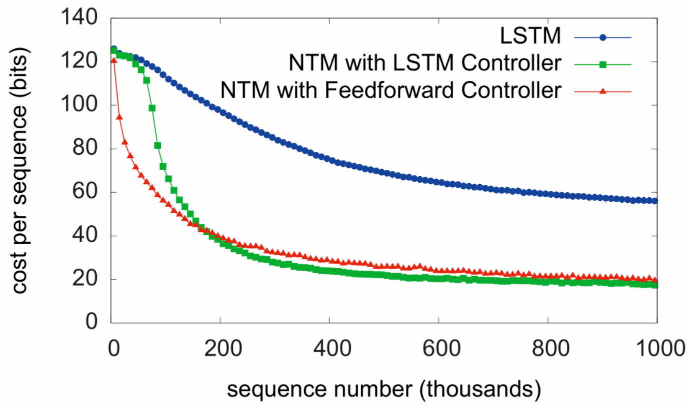
مسئلهٔ رونگاری توانایی ماشین را در نگه‌داری و یادآوری عبارت‌های به دلخواه طولانی بررسی می‌کند. ورودی شبکه دنباله‌ای طولانی از بردارهای دودویی است، که با یک علامت تمام می‌شوند. پس از دریافت علامت ماشین باید بتواند دنباله را دقیقا کپی کند. ورودی‌های آموزشی شبکه هر کدام دنباله‌ای بین ۱ تا ۲۰ عضوی از بردارهایی شامل ۸ بیت داده بودند. خروجی مورد انتظار از ماشین عینا همان دنبالهٔ ورودی (بدون علامت انتهایی) بود. ساختار برنامهٔ مشابه زیر است.

```
move write position to begining of memory
do
    read input
    write input on memory
    increament write position
while input was not delimiter
move read position to begining of memory
while memory data is not delimiter do
    read from memory
    emit output
    increament read position
end
```

نمودار ۴.۳ تفاوت سرعت همگرایی سه روش را به خوبی نشان می‌دهد. همچنین نمودار ۴.۴ نشان می‌دهد که NTM توانسته عملکرد خود را با افزایش طول ورودی در مقایسه با LSTM رو نگاری را با دقت بسیار بالاتری انجام دهد.



شکل ۴.۵: نمونهٔ ورودی و خروجی در مسئلهٔ مرتب‌سازی



شکل ۴.۶: نرخ همگرایی NTM و LSTM در مسئلهٔ مرتب‌سازی

مرتب‌سازی

در این مسئله دنباله‌ای از بردارهای ۸ بیتی به همراه یک مقدار اولویت متناظر با هر بردار به ماشین داده شد. مقدار اولویت یک عدد تصادفی بود که از بازهٔ $[-۱, ۱]$ با احتمال یکنواخت انتخاب می‌شد. در مقابل از ماشین انتظار می‌رفت دنباله‌ای شامل ۱۶ بردار با بیشترین اولویت را به ترتیب اولویت‌شان خروجی دهد. شکل ۴.۵ نمونه‌ای از داده‌های ورودی و خروجی را نشان می‌دهد.

بررسی دسترسی‌های ماشین به حافظه نشان می‌دهد که ماشین ابتدا با استفاده از اولویت، موقعیت نوشتن هر بردار در خروجی را حدس می‌زند و سپس با توجه به ترتیب به دست آمده خروجی‌ها را چاپ می‌کند. نمودار ۴.۶ نشان می‌دهد که NTM در این مسئله نیز توانسته است بهتر از LSTM عمل کند. هر چند NTM برای حل مؤثر این مسئله نیاز به ۸ اشاره‌گر برای خواندن و نوشتن اطلاعات به صورت همزمان داشت که نشان‌گر پیچیدگی بسیار بالای این مسئله است.

۴.۴ الگوریتم‌های تکاملی

«الگوریتم‌های تکاملی»[۳۰] خانوادهٔ دیگری از روش‌های یادگیری هستند که حتی پیش از ابداع ماشین تورینگ به طور غیر مستقیم از آن‌ها صحبت به میان آمده است. کنن[۳۱] در سال ۱۹۳۲ از نظریهٔ انتخاب طبیعی

[۳۰] Evolutionary algorithms
[۳۱] Walter Cannon

به عنوان روشی برای یادگیری نام برد[۲۴]. تورینگ در مقالهٔ مشهور خود علاوه بر معرفی ساختار ماشین محاسبه‌گر از «ارتباط مشخص میان یادگیری ماشین و تکامل» صحبت به میان آورد. فریدمان[۳۲] در سال ۱۹۵۹ پیشنهاد کرد که شبیه‌سازی تکامل از طریق جهش و انتخاب طبیعی می‌تواند منجر به طراحی یک «ماشین هوشمند» شود، و برای اثبات ادعا برنامه‌های کامپیوتری را معرفی نمود که شطرنج بازی می‌کردند.

این الگوریتم‌ها نمایندهٔ خانوادهٔ بزرگی از روش‌های یادگیری به نام روش‌های «یادگیری تقویتی»[۳۳] هستند. این خانواده از یادگیری از داده‌های آموزشی بهره نمی‌برند. در مقابل روش‌های یادگیری در این خانواده توسط یک تابع امتیاز دهی هدایت می‌شوند. به طور کلی الگوریتم‌های این خانواده جواب‌های مختلفی را حدس می‌زنند و با توجه به بازخورد تابع امتیاز دهی، اقدام به بهبود رفتار خود می‌کنند. هر چند این روش‌ها را می‌توان هم در یادگیری با نظارت و هم در یادگیری بدون نظارت به کار برد، ولی معمولا زمانی از این روش‌ها استفاده می‌شود که ما طبقه‌بندی مشخصی برای خروجی مورد انتظار از ماشین نداریم، و صرفا می‌توانیم کیفیت خروجی تولید شده توسط الگوریتم را بررسی کنیم. با این توصیفات می‌توان این خانواده از روش‌های یادگیری را نماینده‌ای از روش‌های یادگیری بدون نظارت دانست.

در راستای بررسی این الگوریتم‌ها ری[۳۴] پژوهشی انجام داد که نتایج آن را در سال ۱۹۹۱ [۲۵] منتشر نمود. او محیطی ایجاد کرد که برنامه‌های کامپیوتری که به زبان اسمبلی نوشته شده‌اند بتوانند به رقابت با یکدیگر بپردازند. برنامهٔ اولیه شامل سه بخش می‌شد: بخش اول که تعداد دستورات برنامه را بر می‌گرداند، بخش دوم که حلقهٔ تولید مثل نام داشت شامل حلقه‌ای بود که برنامه را کپی می‌کرد و در نهایت بخش سوم در بر گیرندهٔ روندی بود که توسط حلقهٔ تولید مثل فراخوانی می‌شد و یک دستور را کپی می‌کرد. حافظهٔ کامپیوتر، پردازنده و سیستم‌عامل محیط شبیه‌سازی را تشکیل می‌دادند. پردازنده به گونه‌ای بین برنامه‌ها تقسیم شده بود که برنامه‌های با طول کوتاه‌تر سریع‌تر بتوانند تولید مثل کنند. همچنین برنامه‌هایی که در حین اجرا با خطا مواجه می‌شدند با توجه به تابعی از تعداد خطاها از لیست برنامه‌ها حذف می‌شدند. خطا نتیجه جهش‌هایی بود که ممکن بود در دستور برنامه‌ها ایجاد شود. این جهش‌ها ممکن بود در هنگام تولید مثل به وجود بیایند و یا در پس زمینه، به صورت تصادفی بر روی هر کدام از برنامه‌ها اتفاق بیفتند.

برنامهٔ اولیه شامل ۸۰ دستور بود که توسط یک برنامه‌نویس نوشته شده بود هستهٔ اولیهٔ شبیه‌سازی را به وجود می‌آورد. با توجه به شرایط گفته شده شبیه‌سازی شروع انجام شد و در طول شبیه‌سازی در نتیجهٔ فرایند تکامل برنامه‌های مختلفی با کارایی‌های متنوع به وجود آمدند. ری گروهی از برنامه‌های به وجود آمده را برنامه‌های انگلی[۳۵] می‌نامد. این دسته برنامه‌ها از روند کپی کردن برنامه‌های دیگر برای تولید مثل خود استفاده می‌کردند، و به این وسیله موفق شده بودند طول خود را کاهش دهند. دسته‌های بسیار متنوع دیگری نیز در این فرآیند تولید شده‌اند که توسط ری توصیف شده‌اند[۲۵]. این شبیه‌سازی نشان داد که

[۳۲] George Friedman
[۳۳] Reinforcement Learning
[۳۴] Thomas Ray
[۳۵] parasitic

رفتارهای پیچیده می‌توانند از دینامیک تکامل بر روی برنامه‌های بسیار ساده به وجود آیند.

۴.۴.۱ تحلیل محاسباتی

روش‌های تکاملی را نیز همانند دیگر روش‌های بهینه‌سازی، می‌توان تحلیل و بررسی کرد. روش‌های تکاملی معمولا شامل روندهای پیچیده و احتمالی غیر خطی است، که بررسی دقیق ریاضی را سخت‌تر می‌کند. در نتیجه در بسیاری موارد برای تحلیل یک روش، مدل ساده شدهٔ آن را بررسی می‌کنند. ولی این ساده‌سازی نیز مشکلی به همراه دارد، مدل ساده‌سازی شده الزاما با مدل اصلی در همهٔ حالات رفتار مشابه ندارد و در نتیجهٔ ساده‌سازی تمام نتایج قابل تعمیم به مدل اصلی نیست. از طرف دیگر برای بررسی رفتار کلی یک الگوریتم تکاملی نیازی نیست جزئیات رفتاری الگوریتم به طور دقیق مورد بررسی قرار بگیرد.

طبعا برای هر تحلیلی ابتدا باید یک مدل دقیق از موضوع مورد مطالعه ارائه شود. الگوریتم‌های تکاملی ساختار بسیار متوعی دارند ولی تقریبا تمامی این الگوریتم‌ها تلاش دارند برداری مانند x، که به آن «ژن»[۳۶] گفته می‌شود، را بیابند به طوری که تابعی مانند $f(x)$ را بهینه کند. در این راستا هر بردار مانند x توسط تابع $f(x)$ ارزش‌گذاری می‌شود. یک الگوریتم تکاملی مجموعه‌ای از بردارها را به طور تصادفی انتخاب می‌کند، سپس بر اساس یک روش انتخاب‌گر زیرمجموعه‌ای از این بردارها را جدا می‌کند و با استفاده از این زیر مجموعه «نسل»[۳۷] بعد را می‌سازد. در تولید نسل بعد معمولا روش‌هایی برای تغییر در تک به تک بردارها به کار می‌رود که منجر به تولید بردارهای جدید در طول فرآیند تکامل شود.

آنچه گفته شد تقریبا تمامی روش‌های تکاملی را در بر می‌گیرد. بردار x می‌تواند هر شیء ریاضیاتی باشد، به عنوان مثال مجموعه‌ای از اعداد حقیقی و یا مجموعه‌ای از اعداد صحیح و یا ترکیبی از این دو. روش‌های انتخابی می‌توانند به صورت حذف دسته‌ای بردارهای با ارزش پایین باشند و یا تولید بردارهای بیشتر از بردارهایی که امتیاز بیشتری داشته‌اند. تولید جواب‌های جدید می‌تواند با ایجاد یک تغییر جزئی در بردار «والد»[۳۸] به وجود بیایند و یا با ترکیب کردن چند بردار والد ایجاد شوند.

همگرایی یکی از ابتدایی‌ترین روش‌های بررسی یک الگوریتم بهینه‌سازی نقطهٔ «همگرایی»[۳۹] آن (در صورت وجود) است. ولی هلند[۴۰] در کتاب خود[۲۶] نشان می‌دهد این معیار، روش خوبی برای ارزش‌گذاری بر توان الگوریتم‌های تکاملی نیست. بسیاری از الگوریتم‌های دیگر مانند گشتن در فضای بسیار بزرگ همهٔ حالات هستند که جواب‌های بسیار بهتری تولید می‌کنند ولی در عمل قابل اجرا نیستند. ولی از طرف دیگر هر الگوریتمی که در نقطهٔ بهینهٔ محلی[۴۱] همگرا نشود به تمام الگوریتم‌هایی که ممکن است در دام

[۳۶] Gene
[۳۷] Generation
[۳۸] Parent
[۳۹] Convergence
[۴۰] Holland
[۴۱] Local optima

نقاط بهینهٔ محلی بیفتند اولویت دارد. اتمار[۴۲] [۲۷] و هلند[۲۶] و بساری دیگر نشان داده‌اند گونه‌هایی از الگوریتم‌های تکاملی وجود دارند که در نقاط بهینهٔ محلی به دام نمی‌افتند.

روش‌های دارای قالب یافتن جواب بهینه معمولا به صورت جست‌وجو در یک فضای حالت بسیار بزرگ انجام می‌شود. این جست‌وجو می‌تواند در بردارهای کامل جواب را بررسی کند، و یا به طریقی بردارهایی منطبق بر یک «قالب»[۴۳] را مورد بررسی قرار دهد. هر چند روش دوم نیاز به وجود یک روش امتیازدهی جزئی دارد، چیزی که در همهٔ مسائل در دسترس نیست. یکی از روش‌های این امتیازدهی جزئی می‌تواند بررسی تمام جواب‌هایی باشد که شامل این بخش مشخص شده هستند. میانگین ارزش تمام جواب‌های می‌تواند یک ناشنگر از کارایی یک بخش خاص باشد. به عنوان مثال فرض کنید فضا جواب‌ها شامل بردارهایی شامل ۳ بیت باشد. ارزش یک جزء جواب مانند $x_۱ = ۱$ برابر میانگین ارزش تمام حالاتی است که مؤلفهٔ اولشان ۱ است. این مجموعه از جواب‌ها را می‌توان به صورت [##۱] نوشت، به طوری که # به معنای مؤلفهٔ نامشخص است. به دسته جواب‌هایی مانند [##۱] یک قالب گفته می‌شود. این روش میانگین توسط فیشر[۴۴] پیشنهاد شده است[۲۸]. هر چند میانگین گرفتن عملگری خطی است، و به جز در مسائلی که امتیاز کلی تابعی خطی از بردار جواب x باشد، کارایی چندانی نخواهد داشت.

روش‌های انتخاب جواب‌های برتر روش‌های بسیاری برای انتخاب ژن‌های برتر وجود دارد. هر کدام از توابع به نوبهٔ خود تلاش می‌کنند که بر روی ژن‌های موفق تاکید بیشتری کنند و ژن‌هایی که عملکرد خوبی ندارند را از مجموعهٔ جواب‌ها حذف کنند. این تاکید می‌تواند مطلق (به عنوان مثال برخی از ژن‌های ناکارآمد حذف شوند) و یا شایسته‌پروری (به ژن‌های با عملکرد بهتر اجازهٔ تولید مثل بیشتری داده شود) باشد. میزان سخت‌گیری یک الگوریتم انتخاب را می‌توان بر اساس تنوع ارزش ژن‌ها پس از اعمال تابع انتخاب بررسی کرد. به طور معمول یک روش انتخابی را با تعداد نسل‌هایی بررسی می‌کنند که ماشین شبیه‌سازی کند تا زمانی که یک ژن تمامی جمعیت ژن‌های مورد بررسی را پوشش دهد. به این زمان در ادبیات الگوریتم‌های تکاملی زمان تصاحب[۴۵] گفته می‌شود.

سرعت همگرایی هرچند اثبات همگرایی کلی روش‌های تکاملی به تنهایی اثباتی بسیار ارزشمند است، ولی بررسی سرعت همگرایی این روش‌های نیز از اهمیت بالایی برخوردار است. افراد بسیاری در مورد سرعت همگرایی روش‌های تکاملی پژوهش انجام داده‌اند و کمک‌های بسیاری به بررسی این شاخه کرده‌اند. از جملهٔ آن‌ها می‌توان به رچنبرگ[۴۶] [۲۹]، باک[۴۷] [۳۰]، بیر[۴۸] [۳۱] و دیگران اشاره کرد. بسیاری از این مقالات در

[۴۲] Atmar
[۴۳] Schema Processing
[۴۴] Fisher
[۴۵] Takeover time
[۴۶] Rechenberg
[۴۷] Bäck
[۴۸] Beyer

مورد الگوریتم‌هایی هستد که یک متغیر پیوسته را بهینه‌سازی می‌کنند و از یک توزیع گسسته برای جهش با استفاده از یک والد استفاده می‌کنند. هر چند همین حالت بسیار محدود نیز از پیچیدگی بسیار بالایی برخوردار است.

۴.۴.۲ الگوریتم‌های تکاملی در عمل

الگوریتم‌های تکاملی در عمل در بسیاری از شاخه‌ها به کار گرفته شده‌اند. مسائلی از قبیل بهینه‌سازی، طراحی هوش مصنوعی برای بازی‌های رایانه‌ای و حتی طراحی بازی‌ها[۳۲] توسط الگوریتم‌های تکاملی بررسی شده‌اند.

فصل ۵

مدل کلی یادگیری

فرض کنید وارد یک جزیرهٔ استوایی شده‌اید. در این جزیره‌ی به خصوص انبه بخش مهمی از رژیم غذایی افراد محلی را تشکیل می‌دهد، ولی شما به عنوان فردی که از یک منطقه‌ی سردسیر هاجرت کرده‌اید اطلاعات کاملی از نحوهٔ تشخیص انبه‌ی خوب از بد ندارید. با این وجود پس از مدت زمانی کوتاه کم کم یاد می‌گیرید که با نگاه به ظاهر یک انبه، میوه‌های خوش طعم را از دیگر میوه‌ها جدا کنید. ذهن شما در طول این فرآیند «آموزش» می‌بیند که با ورودی گرفتن مؤلفه‌هایی همانند رنگ، اندازه و یا سفتی یک انبه طعم آن را پیش‌بینی کند.

۵.۱ مدلی کلی برای یادگیری[۴]

۵.۱.۱ مدل یادگیری آماری

برای تحلیل هر مسئله‌ای در اولین قدم باید مدلی گویا از آن مسئله ارائه دهیم. یک مدل یادگیری آماری از اجزای زیر تشکیل شده است:

- **مجموعه‌ی دامنه**: مجموعه‌ای دلخواه مانند $\mathcal{X}$. این مجموعه شامل تمام اجزایی است که انتظار داریم ماشین بتواند در مواجهه با آن‌ها تصمیم گیری کند. در مثال تشخیص طعم انبه، میوه‌های انبه این مجموعه را تشکیل می‌دهند. البته به طور معمول هر کدام از اعضای مجموعه توسط برداری از اعداد نمایش داده می‌شوند که بردار ویژگی‌های مجموعه نام دارد. در مثال فوق بردار ویژگی‌ها شامل رنگ، اندازه و سفتی میوه‌ها است.

- **مجموعه‌ی برچسب:** مجموعه‌ای دلخواه که خروجی‌های ممکن ماشین را تشکیل می‌دهند. به عنوان مثال اگر بخوایم تشخیص دهیم انبه‌ها مرغوب هستند یا خیر، این مجموعه شامل دو عضوی $\{۰ ۱\}$ خواهد بود که ۰ به معنای انبهٔ نامرغوب و ۱ نشانگر انبهٔ مرغوب است. این مجموعه را با علامت $\mathcal{Y}$ نمایش می‌دهیم.

- **داده‌های آموزش:** مجموعه‌ی $\mathcal{S} = ((x_۱, y_۱), \ldots, (x_n, y_n))$ که دنباله‌ای متناهی از اعضای $\mathcal{X} \times \mathcal{Y}$ است. به عبارت دیگر این مجموعه شامل تعدادی متناهی از نقاط برچسب‌دار دامنه است.

- **نتیجه‌ی فرآیند آموزش:** در نتیجه‌ی فرآیند آموزش تابعی مانند $h : \mathcal{X} \to \mathcal{Y}$ تولید می‌شود که می‌تواند با ورودی گرفتن یک عضو از دامنه برچسب آن را حدس بزند. این تابع اسامی دیگری از جمله تابع طبقه‌بندی، فرضیه و تابع پیش‌بین نیز دارد. برای نشان دادن فرضیهٔ تولید شده توسط الگوریتم A پس از دریافت داده‌های ورودی $\mathcal{S}$ از نماد $A(\mathcal{S})$ استفاده می‌کنیم.

- **تولید داده‌های آموزشی:** فرض می‌کنیم تمام داده‌های آموزشی از توزیعی مانند D تولید شده اند. هر چند الگوریتم آموزش‌گر هیچ اطلاعاتی از ساختار این توزیع ندارد، ولی وجود یک توزیع

جزو پیش‌فرض‌های آموزش است. همچنین فرض می‌کنیم تابعی مانند $f(x_i) = y_i$ وجود دارد که برچسب‌گذاری درست تمام داده‌ها را تولید می‌کند. هدف الگوریتم آموزش‌گر پیش‌بینی رفتار این تابع است.

- **درستی سنجی:** خطای یک فرضیه را برابر احتمال حالتی در نظر می‌گیریم که برچسب گذاری انجام شده توسط فرضیه با برچسب گذاری اولیه توسط تابع f یکسان نباشد. به بیان دقیق‌تر خطا برابر احتمال انتخاب یک عضو از دامنه مانند x است به طوری که $h(x) \neq f(x)$. این خطا را به صورت

$$L_{(D,f)}(h) = D(\{x : h(x) \neq f(x)\})$$

نمایش می‌دهیم. اندیس‌های D و f بر وابستگی تابع خطا به توزیع و تابع ارزش‌گذاری اولیه تاکید دارند.

۵.۱.۲ خطای تجربی

همان‌طور که قبلا گفته شد، یک تابع آموزش‌گر به توزیع اولیهٔ D و یا به تابع ارزش‌گذاری f دسترسی ندارد. پس نمی‌تواند مستقیما خطای نتیجه‌ی آزمایش را بررسی نمود. ولی از طرف دیگر می‌توان خطای فرضیه‌ی $A(S)$ را با استفاده از خود داده‌های S سنجید. به عبارت دقیق‌تر می‌توانیم بنویسم

$$L_S = \frac{|\{i < m : h(x_i) \neq y_i\}|}{m}$$

در حالی که m برابر تعداد نمونه‌های موجود در دنباله‌ی S است. مقدار L_S این عبارت را خطای تجربی می‌نامیم.

هر چند خطای تجربی تنها راه منطقی در مواجهه با مسئله‌ی ارزش‌گذاری نتیجه‌ی یک فرایند آموزش است، ولی در برخی موارد می‌تواند به شکل فاجعه‌آمیزی تخمین غلطی از خطای فرضیه ارائه دهد. به عنوان مثال فرض کنید یک فرضیه برای تمامی داده‌های آموزشی مقدار برچسب و در دیگر موارد مقدار پیش‌فرض ۰ را خروجی دهد. به بیان دیگر داشته باشیم:

$$h_S = \begin{cases} y_i & \exists i \; s.t. \; x_i = x \\ ۰ & \text{در غیر این صورت} \end{cases}$$

هر چند این فرضیه به نظر کاملا ساختگی می‌رسد، ولی ممکن است خروجی یک الگوریتم آموزش‌گر در عمل این چنین تابعی باشد. حال فرض کنید داده‌های آموزشی بخش کوچکی از کل فضای دامنه را تشکیل دهد، و نیمی از داده‌ها طبق تابع ارزش‌گذاری اولیه مقدار ۱ داشته باشند.

۵.۱.۳ یادگیری تقریبا احتمالا درست

در ادامه برای بررسی دقیق‌تر یادگیری مدل یادگیری «تقریبا احتمالا درست»[۱] را معرفی می‌کنیم. این مدل حالت جامع‌تر و دقیق‌تر از مدل یادگیری‌ای است که پیش‌تر ذکر کردیم. مدل یادگیری PAC به ما این اجازه را می‌دهد که به بررسی محدودیت‌های عملیاتی یادگیری بپردازیم.

تعریف ۱.۵. می‌گویم خانواده‌ای از توابع مانند $\mathcal{H}$، «PAC آموزش‌پذیر» هستند هر گاه تابعی مانند $m_{\mathcal{H}} : (۰,۱)^۲ \rightarrow \mathbb{N}$ و یک الگوریتم یادگیری مانند $A_{\mathcal{H}}$ وجود داشته باشد به طوری که در خاصیت زیر صدق کنند.

به ازای هر $\epsilon, \delta \in (۰,۱)$ و به ازای هر توزیع دلخواه D بر روی دامنه‌ی $\mathcal{X}$ و به ازای هر تابع برچسب‌گذاری مانند $f : \mathcal{X} \rightarrow \{۰,۱\}$ اگر بتوان با استفاده از توابع خانواده‌ی $\mathcal{H}$ داده‌ها را تفکیک کرد، آنگاه با اجرا کردن الگوریتم یادگیری بر روی حداقل $m \geq m_{\mathcal{H}}(\epsilon,\delta)$ داده‌ی آموزشی که به طور مستقل از توزیع D استخراج شده‌اند، الگوریتم A به احتمال $۱ - \delta$ فرضیه‌ای مانند h تولید کند به طوری که $L_{(D,f)}(h) \leq \epsilon$

در این تعریف کلمه‌ی «تقریبا» برای توصیف متغیر ϵ و کلمه‌ی «احتمالا» برای توصیف متغیر δ به کار می‌رود. این پارامترها به ترتیب مشخص می‌کنند که دقت جواب‌های تولید شده توسط آموزش‌گر در چه حد است، و آموزش‌گر با چه احتمالی می‌تواند تابعی با آن دقت را خروجی دهد.

تعریف ۲.۵. تابع $m_{\mathcal{H}} : (۰,۱)^۲ \rightarrow \mathbb{N}$ را پیچیدگی یادگیری $\mathcal{H}$ می‌نامیم. این تابع حداقل تعداد نمونه‌های مورد نیاز برای آموزش خانوادهٔ $\mathcal{H}$ را مشخص می‌کند. پیچیدگی یادگیری تابعی از دو متغیر دقت یادگیری ϵ و ضریب اطمینان δ هستند. همچنین این تابع وابسته به خواص خانواده‌ی $\mathcal{H}$ نیز هست.

لازم به ذکر است که اگر $\mathcal{H}$ خانواده‌ای PAC آموزش‌پذیر باشد، توابع بسیاری می‌توانند خاصیت مورد نیاز برای آموزش‌پذیری را ارضا کنند. در میان این توابع ما به دنبال تابعی می‌گردیم که «کمینه» باشد. به عبارت دقیق‌تر داشته باشیم $m_{\mathcal{H}}(\epsilon,\delta)$ حداقل مقدار صحیحی باشد که با دقت ϵ و اطمینان δ می‌تواند خاصیت آموزش‌پذیری PAC را ارضا کند.

گزاره ۳.۵. هر کلاس متناهی مانند $\mathcal{H}$ آموزش‌پذیر PAC است و پیچیدگی یادگیری‌ای از مرتبهٔ زیر دارد:

$$m_{\mathcal{H}} \leq \left\lceil \frac{log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$$

۵.۱.۴ agnostic pac learning

هرچند تعریف فوق از یادگیری PAC می‌تواند مدلی دقیق و قابل بررسی از یادگیری ارائه دهد، ولی مشکلاتی نیز به همراه دارد. یکی از شرایط اولیه‌ی این مدل تفکیک‌پذیر بودن فضای مسئله توسط فضای $\mathcal{H}$ بود. در

[۱]Probably Approximately Correct

بسیاری مسائل اما، تفکیک‌پذیری در شرطی بسیار سختگیرانه است. به عنوان مثال اگر اطلاعات داده شده به الگوریتم یادگیری نتوانند تمامی جوانب مسئله را توصیف کنند، نمیتوانیم خانواده‌ای آموزش‌پذیر برای مسئله ارائه دهیم. این در حالی است که در بسیاری مسائل واقعی ما با واقعیت رخداد مورد بررسی آشنا نیستیم و احتمالا نمی‌توانیم تمامی اطلاعات مشخصه‌های مرتبط با مسئله را ارزیابی کنیم. در برخی موارد نیز، خانوادهٔ $\mathcal{H}$ معرفی شده، ممکن است پیچیدگی لازم را برای توصیف واقعیت فضا نداشته باشد. به عنوان مثال اگر سعی کنیم داده‌هایی که از یک تابع اولیه‌ی غیر خطی تولید شده‌اند را توسط تابعی خطی تقریب بزنیم، همواره خطایی بیش از یک مقدار مشخص خواهیم داشت.

این مشکلات سبب می‌شود که مدل یادگیری Agnostic PAC را معرفی کنیم. در این مدل با ضعیف کردن شرط دقت یک الگوریتم یادگیری اجازه می‌دهیم که خطای آن به مقدار مشخصی افزایش بیابد. به عبارت دقیق‌تر خطای یک الگوریتم را در مقایسه با بهترین تابع موجود در خانوادهٔ $\mathcal{H}$ را می‌سنجیم. برای این منظور ابتدا توزیع D را گسترش می‌دهیم،به طوری که توزیعی بر روی فضای $X \times Y$ را توصیف کند. این گسترش به ما اجازه می‌دهد شرایطی را که فضای ورودی نمی‌تواند به درستی اطلاعات مورد نیاز برای تابع هدف را توصیف کند در بر بگیریم. در ادامه تعریف خطا را با توجه به تعریف جدید توزیع D گسترش می‌دهیم. تعریف می‌کنیم

$$L_D(h) = \underset{(x,y)\sim D}{\mathbb{P}}[h(x) \neq y] = D(\{(x,y) : h(x) \neq y\})$$

در مدل Agnostic PAC ما به دنبال تابعی مانند $h \in \mathcal{H}$ هستیم به طوری که مقدار خطا کمینه شود. به عبارت دقیق‌تر می‌گوییم اگر به ازای توزیعی مانند D این خطا در تابعی مانند h_D کمینه شود، ما باید بتوانیم توابعی مانند h را بیابیم به طوری که $L_D(h) \leq L_D(h_D) + \epsilon$. در ادامه تعریف پیچیدگی یادگیری را نیز به تناسب تغییر می‌دهیم. به عبارت دقیق‌تر خواهیم داشت:

تعریف ۴.۵. می‌گوییم خانواده‌ای از توابع مانند $\mathcal{H}$، «*Agnostic PAC* آموزش‌پذیر» هستند هر گاه تابعی مانند $m_{\mathcal{H}} : (۰,۱)^۲ \to \mathbb{N}$ و یک الگوریتم یادگیری مانند $A_{\mathcal{H}}$ وجود داشته باشد به طوری که در خاصیت زیر صدق کنند.

به ازای هر $\epsilon, \delta \in (۰,۱)$ و به ازای هر توزیع دلخواه D بر روی دامنه‌ی $\mathcal{X}$ و به ازای هر تابع برچسب‌گذاری مانند $f : \mathcal{X} \to \{۰,۱\}$ اگر تابع $h_D \in \mathcal{H}$ از توابع خانواده‌ی $\mathcal{H}$ کمترین خطا را داشته باشد، آنگاه با اجرا کردن الگوریتم یادگیری بر روی حداقل $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ داده‌ی آموزشی که به طور مستقل از توزیع D استخراج شده‌اند، الگوریتم A به احتمال $۱ - \delta$ فرضیه‌ای مانند h تولید کند به طوری که $L_D(h) \leq L_D(h_D) + \epsilon$

۵.۱.۵ قلمروهای دیگر در یادگیری

هر چند هر آنچه تا این‌جا گفته شد، محدود به یادگیری یک برچسب‌گذاری دو ارزشی بود،ولی هیچ دلیلی وجود ندارد که تمام الگوریتم‌های یادگیری را به مدل محدود کنیم. می‌توان مدل یادگیری PAC را گسترش داد تا قلمروهای دیگری از عملیات‌های یادگیری را پوشش دهد.

- **دسته‌بندی با بیش از دو کلاس** دسته‌بندی‌های انجام شده توسط یک الگوریتم یادگیری الزاما به دو کلاس خوب یا بد محدود نمی‌شوند. به عنوان مثال فرض کنید می‌خواهیم الگوریتمی را آموزش دهیم که با خواندن یک مقاله از یک روزنامه، موضوع مقاله را تشخیص دهد. موضوع یک مقاله می‌تواند اقتصادی، علمی، ورزشی و یا هنری باشد. در این مثال مجموعه‌ی دامنه شامل تمام مقالات نوشته شده است و مجموعه‌ی برچسب‌ها را عبارات اقتصادی، علمی، ورزشی و هنری تشکیل می‌دهد.

- **برآوردگر** برخی مسائل یادگیری پا را از دسته‌بندی فراتر می‌گذارند و به دنبال توابعی از فضا صحیح به فضای صحیح می‌گردند. در این مسائل ما به دنبال یافتن یک الگو در میان داده‌های ورودی هستیم. به عنوان مثال فرض کنید در مسئله‌ای می‌خواهیم وزن یک کودک را هنگام تولد پیش از تولد او تخمین بزنیم. داده‌های ورودی این مسئله می‌توانند اطلاعاتی باشند که از یک آزمایش سونوگرافی حالی می‌شود: قطر سر، قطر بدن، طول بدن. در این مسئله فضای دامنه‌ی ما شامل بردارهایی سه بعدی از اعداد حقیقی است و برچسب‌ها خود نیز هر کدام عددی حقیقی هستند. در این حالت خاص بهتر از تابع دیگری برای یافتن میزان موفقیت یادگیری بهره گیریم که نزدیکی برآورد را نیز بازگو کند. به طول دقیق‌تر تعریف می‌کنیم:

$$L_D(h) = \mathop{\mathbb{E}}_{(x,y)\sim D}(h(x)-y)^{۲}$$

۵.۲ یادگیری تدریجی

بسیاری از الگوریتم‌های موجود، یادگیری را در مراحل قابل تفکیک و به صورت تدریجی انجام می‌دهند. از طرف دیگر در فصل‌های قبل مدل‌های محاسباتی متعددی معرفی شدند، که می‌توان یادگیری را در هر کدام از این مدل‌های محاسباتی به صورت جداگانه بررسی نمود. در این بخش قصد داریم مدلی ارائه کنیم که بتواند ابزار مورد نیاز برای تحلیل محاسباتی یادگیری به صورت تدریجی را توصیف کند.

با وجود تعدد مدل‌های محاسباتی بسیاری از آن‌ها در سه خاصیت اساسی مشترک هستند:

- یک ماشین محاسبه‌گر ، توصیفی متناهی دارد.
- مراحل مختلف محاسبه توسط ماشین قابل تفکیک هستند.
- در هر مدل محاسباتی ماشینی جهانی وجود دارد که تمام ماشین‌های آن مدل را شبیه‌سازی می‌کند.

این خواص به ما اجازه می‌دهد که مدلی برای یادگیری ماشین ارائه دهیم. مدل معرفی شده، صرفا به بررسی یادگیری ماشین و توانایی‌ها و محدودیت‌های آن می‌پردازد و برای مدل‌سازی محاسبه از دیگر مدل‌های موجود کمک می‌گیرد. آنچه در این فصل بیان می‌شود محدود به هیچ مدل محاسباتی خاصی نیست و می‌توان آن را دربارهٔ انواع مدل‌های محاسباتی بیان نمود. هر چند به وضوح انتظار می‌رود که یک مدل محاسباتی شرایط اولیه ذکر شده را دارا باشد.

۵.۲.۱ تعریف

تعریف ۵.۵. فرض کنید $\mathcal{X}$، $\mathcal{Y}$ دو فضای متریک باشند. همچنین فرض کنید $\mathcal{H} \subset \Sigma^*$ مجموعه‌ای از ماشین‌های محاسبه‌گر باشند که توابعی تام، از فضای $\mathcal{X}$ به فضای $\mathcal{Y}$ را توصیف می‌کنند. همچنین فرض کنید D توزیعی بر روی فضای $\mathcal{X} \times \mathcal{Y}$ باشد. در نهایت برای تابع دلخواه $h \in \mathcal{H}$ مقدار خطا به صورت زیر محاسبه می‌گردد:

$$error_D(h) = \underset{(x,y)\sim D}{\mathbb{P}}(h(x) - y)$$

تعریف ۶.۵. فرض کنید $\mathsf{m}_۱, \mathsf{m}_۲ \in \mathcal{H}$. تابع

$$\mathsf{d}(\mathsf{m}_۱, \mathsf{m}_۲) = \int_{\mathcal{X}} |\lfloor m_۱ \rfloor(x) - \lfloor m_۲ \rfloor(x)| dx$$

یک متر $L_۱$ بر روی فضای توصیفات ماشین‌ها است.

در صورتی که فضای حالت $\mathcal{X}$ فضایی گسسته باشد، انتگرال نوشته شده در عبارت بالا با جمع جایگزین می‌شود.

تعریف ۷.۵. تابع جزئی $\mathsf{t}(data, n) : (\mathcal{X} \times \mathcal{Y})^* \times \mathbb{N} \to \Sigma^*$ به همراه تابع $time : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ یک تابع همگرا می‌نامیم، هرگاه شرایط زیر برقرار باشند:

- اگر تابع t برای ورودی‌هایی مانند $(data, n)$ جوابی مانند $\mathsf{t}(data, n)$ خروجی دهد، داشته باشیم: $\mathsf{t}(data, n) \in \mathcal{H}$.

- این تابع توسط ماشینی مانند $\lfloor t \rfloor$ قابل محاسبه باشد، به طوری که ماشین $\lfloor t \rfloor$ بتواند محاسبهٔ $\mathsf{t}(data, n)$ را در کمتر از $time(|data|, n)$ گذر به پایان برساند. به $\lfloor t \rfloor$ ماشین متناظر با t می‌گوییم.

- در صورتی که تابع در نقطه‌ای مانند $(data, n)$ تعریف شده باشد، در نقطهٔ دیگری مانند $(data, m)$ نیز تعریف شده باشد، به طوری که $m > n$.

- با افزایش n مقدار خروجی تابع به توصیف یک ماشین مشخص همگرا شود، به عبارت دیگر

$$\forall data \exists l_{data}, \lim_{n\to\infty} t(data, n) = l_{data}$$

تعریف ۸.۵. یک تابع همگرا را استاندارد می‌نامیم هرگاه به ازای هر دادهٔ ورودی مانند $data$، یا $\mathsf{t}(data, n)$ در تمامی بازهٔ $n \in \mathbb{N}$ تعریف نشده باشد، و یا عددی مانند σ_{data} وجود داشته باشد به طوری که:

$$\forall data : \quad \forall n < \sigma_{data} \qquad \mathsf{t}(data, n) \uparrow$$

$$\forall data : \quad \forall n \geq \sigma_{data} \qquad \mathsf{t}(data, n) \downarrow$$

استاندارد سازی توابع همگرا به ما این اجازه را می‌دهد که در ادامه بتوانیم مفهوم نسل و معیارهای پیچیدگی یک الگوریتم همگرا را تعریف کنیم.

لم ۹.۵. به ازای هر تابع همگرا مانند t، یک تابع همگرای استاندارد مانند t_s وجود دارد به طوری که اگر تابع t به ازای ورودی $data$ به ماشین l_{data} میل کند، تابع t_s نیز به همان ماشین همگرا شود.

اثبات: هر ماشین همگرا خود یک ماشین محاسبه‌گر است. طبق فرض اولیه هر ماشین محاسبه‌گر دارای یک ماشین جهانی از کلاس خود آن ماشین است. همچنین می‌دانیم با استفاده از ماشین جهانی $[U_c]$ می‌توان ماشین محاسبه‌گر را تا تعداد گام دلخواه شبیه‌سازی کرد. ماشین جهانی خود می‌تواند تعداد گام‌هایی را که پیش رفته است در حافظهٔ خود نگه دارد، و همچنین می‌تواند پیش‌بینی کند برای شبیه‌سازی یک گام بیشتر از ماشین شبیه‌سازی شده، به چقدر زمان نیاز دارد.

با توجه به آنچه گفته شد، ماشین جهانی می‌تواند یک ماشین همگرا را شبیه‌سازی کند و پس از استفاده از تمام زمان خود، آخرین خروجی ماشین شبیه‌سازی شده را چاپ کند. طبعا در صورتی که ماشین همگرا هیچ خروجی‌ای چاپ نکند ماشین جهانی نیز خروجی‌ای نخواهد داد. این ماشین جهانی به واسطهٔ الگوریتم کلی‌ای که گفته شد، ساختمان کلی یک ماشین استاندارد را تشکیل می‌دهند.

لازم به ذکر است که ماشین استاندارد معرفی شده احتمالا نمی‌تواند هم‌سرعت با ماشین شبیه‌سازی شده محاسبه انجام دهد، در نتیجه در صورتی که تابع زمان ماشین استاندارد را تغییر ندهیم ممکن است دو ماشین با دریافت ورودی‌های یکسان خروجی های متفاوتی ارائه دهند. برای تصحیح این مشکل می‌توان برای ماشین استاندارد تابع زمان جدیدی معرفی کرد که کاهش سرعت کارکرد ناشی از استفاده از ماشین جهانی $[U_c]$ را پوشش دهد. □

با توجه به لم ۹.۵ بدون از دست دادن کلیت مسئله از این تابع همگرای استاندارد را به اختصار تابع همگرا می‌نامیم.

تعریف ۱۰.۵. به ازای تابع همگرا t رابطهٔ هم‌ارزی هم‌نسل بودن به صورت زیر تعریف می‌گردد:

$$i \equiv_{data} j \;\Leftrightarrow\; \forall k \in [i, j] : \mathsf{t}(data, i) = \mathsf{t}(data, k)$$

همچنین فرض کنید $i < j \in \mathbb{N}$ دو نسل متوالی برای دادهٔ ورودی $data$ باشند یعنی به بیان ریاضی داشته باشیم $\exists k : i \equiv_{data} k, k + ۱ \equiv_{data} j$. فاصلهٔ میان دو نسل i و j به صورت

$$d_{data}(i, j) = \mathsf{d}(t(data, i), t(data, j))$$

تعریف می‌گردد.

برای بررسی پیچیدگی محاسباتی یک ماشین همگرا مؤلفه‌های مختلفی را می‌توان بررسی نمود که هر کدام به تنهایی بخشی از تصویر کلی را می‌توانند ارائه دهند.

تعریف ۱۱.۵. برای یک ماشین همگرا، پیچیدگی تغییرات تابعی از طول داده‌های ورودی و برابر مقدار بزرگترین اختلاف میان دو نسل متوالی در طی فرآیند محاسبه است در میان تمام داده‌های ورودی ممکن با طول مشخص است.

$$\begin{aligned} complexity_{change}(n) = \quad & max(d_{data}(i, i+1)) \\ s.t. \quad & |data| = n \end{aligned}$$

تعریف ۱۲.۵. برای یک ماشین همگرا، پیچیدگی زمان همگرایی تابعی از طول داده‌های ورودی به ماشین است و برابر بیشترین تعداد گذرهایی است که ماشین در طی فرآیند همگرایی برای عبور از یک نسل طی می‌کند. این معیار پیچیدگی نیز تابعی از طول ورودی است و در میان تمام داده‌های ورودی ممکن با طول مشخص بررسی می‌شود.

$$\begin{aligned} complexity_{time}(n) = \quad & max(time(|data|, j) - time(|data|, i)) \\ s.t. \quad & |data| = n \\ & i \equiv_{data} (j-1) \\ & i \not\equiv_{data} (j) \end{aligned}$$

در بسیاری از الگوریتم‌هایی موجود در دنیای واقعی، توابع همگرا فضای ورودی و خروجی متناهی دارند. در نتیجه پیچیدگی تغییرات نیز می‌تواند حداکثر مقداری متناهی اختیار کند. هر چند این محدودیت ممکن است کارایی این معیار را تا حدودی مورد سوال قرار دهد، ولی بررسی این پارامتر می‌تواند اطلاعات کامل‌تری راجع به نحوهٔ عملکرد الگوریتم و پیشبینی‌پذیری رفتار آن به ما بدهد.

تعریف ۱۳.۵. برای یک ماشین همگرا نرخ همگرایی برابر بیشترین مقدار عبارت زیر در میان تمام ورودی‌های ممکن با طول مشخص است:

$$\begin{aligned} rate(n) = \quad & max \lim_{i\to\infty} \frac{\mathtt{d}(t(data, i+1), \mathtt{l}_{data})}{\mathtt{d}(t(data, i), \mathtt{l}_{data})} \\ s.t. \quad & |data| = n \\ & i \not\equiv_{data} i+1 \end{aligned}$$

که در $\mathtt{l}_{data}$ برابر مقداری است که تابع همگرا به ازای ورودی $data$ به آن میل خواهد کرد.

تعریف ۱۴.۵. برای یک ماشین همگرا مانند t دقت همگرایی برابر خطای ماشین به ازای داده $data$ برابر مقدار تابع خطا پس از g نسل است. به عبارت دیگر :داریم

$$precision(data, g) = error_D(t(data, g))$$

تعریف ۱۵.۵. برای یک ماشین همگرا صحت خروجی تولید شده توسط ماشین پس از g نسل برابر احتمال تولید خروجی با دقت همگرایی بهتر از ϵ در میان تمام داده‌های آموزشی به طول مشخص است. به عبارت دیگر داریم

$$\begin{aligned} accuracy(n, \epsilon, g) = \ & \mathbb{P}precision(data, g) < \epsilon \\ s.t. \qquad & |data| = n \\ & \forall i : data_i \sim D \end{aligned}$$

نمونه‌های موجود در یک دنباله‌ی ورودی مانند $data$ به صورت مستقل از توزیع D استخراج می‌شوند.

تعریف ۱۶.۵. به یک کلاس از توابع مانند $\mathcal{H}$ مجموعه‌ای آموزش‌پذیر تدریجی می‌گوییم اگر به ازای هر توزیع مانند D، تابعی مانند $m_D : (۰, ۱)^۲ \to \mathbb{N}^۲$ وجود داشته باشد به طوری که اگر $m_D(\delta, \epsilon) = (p, q)$ داشته باشیم $accuracy(i, \epsilon, g) > \delta \forall i > p, g > q$.

به مقدار m_D پیچیدگی آموزش تدریجی می‌گوییم.

قضیه ۱۷.۵. کلاس $\mathcal{H}$ آموزش‌پذیر تدریجی است اگر و تنها اگر مطابق تعریف یادگیری تقریبا احتمالا درست آموزش‌پذیر باشد.

اثبات: اثبات این قضیه را در دو بخش انجام می‌دهیم. ابتدا ثابت می‌کنیم هر مجموعه‌ی آموزش‌پذیر تقریبا احتمالا درست آموزش‌پذیر تدریجی نیز هست. سپس نشان می‌دهیم طرف دیگر قضیه نیز درست است.

بخش ۱. فرض کنید کلاس $\mathcal{H}$ آموزش‌پذیر باشد. این تابع توسط الگوریتمی مانند $A_{\mathcal{H}}$ آموزش داده می‌شود. در نتیجه خواهیم داشت $A_{\mathcal{H}}(data) = h$. به سادگی می‌توان تابعی مانند $A'_{\mathcal{H}} : X^* \times \mathbb{N}\Sigma^*$ نوشت که با نادیده گرفتن ورودی دوم خود مقدار $A_{\mathcal{H}}$ را حساب می‌کند. به سادگی می‌توان نشان داد A' یک تابع آموزشگر تدریجی است، و پیچیدگی آموزش تدریجی عملا برابر پیچیدگی آموزش PAC خواهد بود.

بخش ۲. از طرف دیگر فرض کنید کلاس $\mathcal{H}$ آموزش‌پذیر تدریجی باشد. در نتیجه برای این کلاس به ازای هر توزیع دلخواه تابع آموزشگری مانند $A'(data, n)$ وجود دارد. همچنین پیچیدگی آموزش تدریجی نیز تابعی مانند $m'(\delta, \epsilon)$ وجود دارد. تابع $A(data) = A'(data, m'(\delta, \epsilon)_۲)$ یک تابع آموزش‌گر با توجه به تعریف یادگیری تقریبا احتمالا درست خواهد بود. پیچیدگی یادگیری نیز برابر $m(\delta, \epsilon) = m'(\delta, \epsilon)_۱$ خواهد بود.

□

۵.۲.۲ مثال

در الگوریتم‌های یادگیری که امروزه به طور گسترده استفاده می‌شوند، معمولا ماشین‌های آموزش‌پذیر شامل دو بخش هستند: بخشی که از پیش توسط یک برنامه‌نویس مشخص شده است، و بخش دومی که توسط یک الگوریتم یادگیری مقداردهی می‌شود. به این دو بخش به ترتیب، ثابت‌ها و متغیرهای یک الگوریتم آموزش‌پذیر می‌گوییم.

شبکهٔ عصبی چند لایه

یک شبکهٔ عصبی چند لایه، متشکل از چندین لایه نورون است که هر کدام توسط اتصالی با وزن مشخص به تمام نورون‌های لایهٔ بعدی خود متصل است. این شبکه‌ها نمونه‌ای از الگوریتم‌های یادگیری با ناظر هستند. آموزش شبکهٔ عصبی چند لایه را می‌توان به خوبی در قالب یک تابع آموزشگر توصیف نمود. روش پس انتشاری یک از روش‌های معمول برای آموزش شبکه‌های عصبی است که ما در این مثال از آن استفاده خواهیم نمود. با توجه به ساختار این روش یادگیری، ماشین آموزش‌گر و ماشین‌های آموزش‌پذیر بهتر از با مدل محاسبه بلام توصیف شوند. در نهایت به بررسی پیچیدگی‌های این روش آموزشی می‌پردازیم.

همان‌طور که گفته شد، شبکهٔ عصبی متشکل از تعدادی لایه‌های نورون است که به واسطهٔ یال‌هایی با وزن‌های مشخص به یکدیگر متصل شده‌اند. با تغییر وزن یال‌ها در این شبکه خانوادهٔ تمام شبکه‌های عصبی ممکن به وجود می‌آیند. به طور معمول ورودی‌های یک شبکه آرایه از اعداد صحیح در بازهٔ $[-۱, ۱]$ هستند و خروجی‌های شبکه را نیز، آرایه‌ای از اعداد صحیح از همان بازه تشکیل می‌دهند. تعداد عناصر آرایهٔ ورودی و آرایهٔ خروجی معمولا پیش از آموزش معین می‌گردد، و فرآیند آموزش تغییری در ساختار شبکه نمی‌تواند بدهد. به عبارت دیگر ثابت‌های یک ماشین آموزش‌پذیر شامل تعداد لایه‌های شبکهٔ عصبی، توابع فعال‌سازی نورون‌ها و تعداد نورون‌ها در هر لایه می‌شود، و وزن یال‌ها متغییرها الگوریتم را تشکیل می‌دهد. در نتیجهٔ آنچه گفته شد، فضای ورودی توابع آموزش‌پذیر ($\mathcal{S}$)) فضای $[-۱, ۱]^n$، فضای خروجی ($\mathcal{D}$) برابر $[-۱, ۱]^m$ و فضای توابع آموزش‌پذیر ($\mathcal{L}$) در این مثال برابر تمام توابع مانند $f : [-۱, ۱]^n \rightarrow [-۱, ۱]^m$ است که توسط یک ثابت‌های یکسانی دارند و تنها مقادیر متغیرها متفاوت هستند.

شبکهٔ اولیه در مرحلهٔ اول، ماشین آموزشگر در مدت σ مرحله می‌تواند یک شبکهٔ عصبی اولیه را شامل ثابت‌های مشخص و متغیرهای تصادفی، بدون هیچ گونه آموزشی تولید کند و خروجی دهد. این مرحله کاملا مستقل از داده‌های آموزش انجام می‌شود، و در نتیجه مقدار σ عددی مستقل از داده‌های ورودی است. این شبکهٔ عصبی مبنای ادامهٔ کار خواهد بود.

آموزش ماشین آموزشگر مقدار خروجی شبکهٔ عصبی را برای یکی از نمونه‌های ورودی، به عنوان مثال نمونهٔ n ام، برآورد می‌کند. سپس مقدار $e_j(n) = d_j(n) - y_j(n)$ را اندازه می‌گیرد. در این رابطه $d_j(n)$ مقدار خروجی مورد انتظار از شبکه به ازای ورودی n ام است. هدف کاهش خطای شبکهٔ عصبی به بیشترین

مقدار ممکن است. ماشین آموزش‌گر می‌تواند مقدار خطای کلی شبکه به ازای ورودی n ام به صورت زیر قابل محاسبه کند:

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$$

سپس با استفاده از روش بیشترین کاهش، می‌توان تغییر لازم به ازای هر کدام از اتصال‌ها را به روش زیر به دست آورد.

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

در عبارت بالا، مقدار i برای تمام نورون‌های لایهٔ قبلی، و مقدار j برای تمام نورون‌های لایهٔ فعلی تغییر می‌کند. همچنین y_i مقدار خروجی نورون i از لایهٔ قبلی است. در نهایت η مقداری است که سرعت یادگیری نام دارد. این عددی است که مستقل از داده‌های نمونه خارج از فرآیند آموزش انتخاب می‌گردد. انتخاب این مؤلفه به گونه‌ای است که یادگیری از سرعت نسبتا بالایی برخوردار باشد، و در عین حال از نوسان در خروجی جلوگیری شود. مشتق جزئی در تابع بالا خود وابسته به v_j است که خروجی تابع فعال‌سازی نورون j ام است. می‌توان برای نورون‌های خروجی نشان داد که مقدار مشتق از رابطهٔ زیر قابل محاسبه است:

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n)\phi'(v_j(n))$$

در عبارت بالا ϕ' برابر مشتق تابع فعال‌سازی نورون است که تابعی مشخص است. برای نورون‌های میانی محاسبهٔ مقدار مشتق امری به مراتب پیچیده‌تر است، ولی در نهایت می‌توان این مقدار را به صورت زیر ساده نمود:

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = \phi\ (v_j(n))\sum_k -\frac{\partial \mathcal{E}(n)}{\partial v_k(n)} w_{kj}(n)$$

که در این‌جا اندیس k معرف نورون‌های لایه‌های بعدتر از لایهٔ مورد بررسی است. در نتیجه برای تصحیح یک لایه لازم است ابتدا تمام لایه‌های بعدی تصحیح شده باشند.

آن‌چه گفته شد تماما کارهایی است که توسط یک ماشین آموزشگر انجام می‌گردد. خروجی ماشین آموزش‌گر یک تابع بر اساس مدل شبکهٔ عصبی خواهد بود که تمامی پارامترهای آن متغییر معیین شده است.

تحلیل

پیچیدگی زمانی با توجه به آنچه گفته شد، اگر یک آموزش‌گر شبکهٔ عصبی تعداد محدود و ثابتی از نمونه‌ها را در هر گام از آموزش خود مورد بررسی قرار دهد پیچیدگی زمانی آن عددی ثابت خواهد شد. در غیر این صورت پیچیدگی زمانی تابعی خطی از تعداد نمونه‌های ورودی، و در نتیجه طول ورودی خواهد بود.

پیچیدگی تغییرات همچنین با توجه به ساختار روش یادگیری شبکه‌های عصبی به سادگی می‌توان نشان داد که بیشترین مقدار تغییرات ماشین، تنها زمانی می‌تواند اتفاق بیفتد که ماشین برای اولین بار با یک نمونهٔ ورودی مواجه می‌گردد. در مواجهه‌های بعدی میزان خطا قطعا کمتر خواهد بود و در نتیجه میزان تغییرات کمتر می‌شود.

نرخ همگرایی در نهایت نرخ همگرایی تابعی از پارامتر η خواهد بود.

۵.۲.۳ آموزش‌گر جهانی

همانند ماشین‌های محاسبه‌گر در نظریهٔ محاسبه، می‌توان ماشین‌های آموزشگر فضای $\mathcal{S} \to \mathcal{D}$ را نیز با استفاده از اطلاعات متناهی به صورت یکتا مشخص نمود.در نتیجه این سوال به صورت طبیعی مطرح می‌گردد که آیا می‌توان ماشین جهانی آموزشگر نیز تعریف کرد؟ پیچیدگی‌های این ماشین در مقایسه با ماشین‌های شبیه‌سازی شده چقدر است؟ به بیان دقیق‌تر آیا می‌توان یک ماشین آموزشگر جهانی مانند ماشین $[U_t]$ معرفی نمود که داده‌هایی شامل توصیف یک ماشین آموزشگر، و داده‌های آموزشی آن ماشین آموزش‌گر را ورودی بگیرد و خروجی متناظر با آن ماشین چاپ نماید؟

قضیه ۵.۱۸. برای تمام ماشین آموزشگر که دارای تابع زمان مانند $time(n)$ هستند، ماشینی جهانی مانند $[U_t]$ وجود دارد که تابع زمانی از مرتبهٔ $time'(n)$ دارد. نسبت این دو تابع با توجه به افت سرعت ناشی از شبیه‌سازی ماشین محاسبه‌گر قابل اندازه‌گیری است. همچنین کلاس ماشین‌های آموزشگری که در پیچیدگی تغییرات، و یا همگرایی محدودیت‌هایی دارند نیز، شامل یک ماشینی جهانی برای شبیه‌سازی اعصای همان کلاس است.

مثال

فرض کنید تمامی محاسبات آموزشی بر روی ماشین‌های تورینگ دارای دو نوار حافظه انجام می‌شود. مجموعهٔ تمام ماشین‌های آموزشگری را در نظر بگیرید در هر نسل خطای خود را نصف می‌کنند، و از تابع زمان خطی بهره می‌برند. به عبارت دیگر نرخ همگرایی برابر ۰٫۵ دارند و تابع زمان آن‌ها مقداری برای $time(n) = \mathsf{c}.n$ دارد. یک ماشین آموزشگر جهانی وجود دارد که می‌تواند تمامی این ماشین‌ها را شبیه‌سازی کند. همچنین این ماشین نرخ همگرایی برابر ۰٫۵ دارد و از تابع زمانی $time_U(n) = \mathsf{c}.n.log(n)$ برخوردار است.

اثبات: برای اثبات این قضیه ابتدا نیاز داریم بر روی یک مدل محاسبهٔ خاص توافق کنیم. این مدل محاسبه می‌تواند مدل محاسبهٔ اعداد حقیقی، ماشین‌های تورینگ، ماشین‌های تعاملی و یا هر مدل محاسبهٔ دیگری باشد به شرطی که خواص زیر را ارضا کند:

- هر ماشین توصیفی متناهی داشته باشد.
- یک ماشین جهانی نیز در همان مدل محاسبه وجود داشته باشد.
- تابعی مانند $slow(n)$ وجود داشته باشد، به طوری که اگر ماشین شبیه‌سازی شده محاسبه‌ای را در n گام بتواند انجام دهد، ماشین جهانی همان محاسبه را در $slow(n)$ گام به پایان برساند.

به عنوان مثال برای ماشین‌های تورینگ دارای حداقل دو نوار حافظه ثابت شده است ماشین جهانی‌ای وجود دارد که می‌تواند هر محاسبه‌ای n گامی را در $n.log(n)$ شبیه‌سازی کند.[۵، صفحات ۳۲-۲۹]

ابتدا نشان می‌دهیم ماشین جهانی مانند $\lfloor \mathsf{U} \rfloor$ وجود دارد و در ادامه نشان می‌دهیم این ماشین خواص پیچیدگی تغییرات و ترخ همگرایی ماشین شبیه‌سازی شده را حفظ می‌کند.

فرض کنید تابع t یک تابع آموزشگر باشد که توسط ماشین $\lfloor \mathsf{t} \rfloor$ محاسبه می‌گردد. مطابق فرض اولیه، این ماشین دارای یک توصیف متناهی است و ماشین جهانی‌ای مانند $\lfloor \mathsf{U} \rfloor$ وجود دارد که می‌تواند رفتار این ماشین را شبیه‌سازی کند. این ماشین برای شبیه‌سازی n گام از $\lfloor \mathsf{t} \rfloor$ نیاز $slow(n)$ زمان خواهد داشت. پس اگر تابع $t(data,n)$ توسط ماشین $\lfloor \mathsf{n} \rfloor$ در $time(n)$ گذر محاسبه شود، ماشین جهانی می‌تواند خروجی مشابهی در $time'(n) = slow(time(n))$ گذر محاسبه کند.

با توجه به تعریف ماشین جهانی به وضوح خواهیم داشت:

$$\lfloor \mathsf{U} \rfloor (pack(\mathsf{t}, data), n) = \lfloor \mathsf{t} \rfloor (data, n) = t(data, n)$$

در عبارت بالا $pack(a,b) : \Sigma^*\Sigma^* \to \Sigma^*$ تابعی یک به یک است.

بخش ۱. ابتدا نشان می‌دهیم ماشین $\lfloor \mathsf{U} \rfloor$ خود یک ماشین آموزشگر است. فرض کنیم ماشین برای ورودی $(pack(\mathsf{t}, data), n)$ خروجی‌ای تولید کند. خواهیم داشت

- خروجی این ماشین با خروجی ماشین $t(data,n)$ برابر است، پس این خروجی توصیف یک ماشین آموزش‌پذیر است.
- طبق تعریف این ماشین در زمان $time'(n)$ خروجی را می‌تواند محاسبه نماید.

- با توجه به اینکه t خود یک ماشین آموزش‌گر است، پس به ازای نقطهٔ دیگری مانند $(data, m)$ نیز خروجی چاپ خواهد نمود. و در نتیجه ماشین جهانی نیز به ازای $\lfloor \mathsf{U} \rfloor (pack(\mathsf{t}, data), m)$ خروجی چاپ خواهند نمود.

- با توجه به همگرایی تابع شبیه‌سازی شده، ماشین جهانی نیز به ازای هر دادهٔ دلخواه همگرا خواهد بود.

بخش ۲. خروجی‌های ماشین $\lfloor U \rfloor$ دقیقا معادل خروجی‌های ماشین‌های شبیه‌سازی شده است. در نتیجه دو ماشین به ازای ورودی‌های یکسان خروجی‌های یکسان تولید می‌نمایند. در نتیجه اگر در خروجی ماشین جهانی تغییری بین دو نسل ایجاد شود، معادل همین تغییر را در ماشین شبیه سازی شده خواهیم دید و برعکس.

$\square$

توجه کنید، در اثبات بالا فرض بر این است که کد ماشین داده شده به ماشین جهانی یک کد صحیح از یک ماشین آموزش‌گر است. در غیر این صورت هیچ تضمینی در مورد نحوهٔ عملکرد ماشین داده نمی‌شود.

کتاب‌نامه

[1] Sonderegger, Dina Q. Goldin; Scott A. Smolka; Paul C. Attie; Elaine L. Turing machines, transition systems, and interaction. *Information and Computation*, 194, 2004.

[2] Blum, Lenore, Shub, Mike, and Smale, Steve. On a theory of computation and complexity over the real numbers: np- completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc. (N.S.)*, 21(1):1–46, 07 1989.

[3] Leeuwen, Jan Van and Wiedermann, Jirí. The turing machine paradigm in contemporary computing. in *Mathematics Unlimited - 2001 and Beyond. LNCS*, pp. 1139–1155. Springer-Verlag, 2000.

[4] Shalev-Shwartz, Shai and Ben-David, Shai. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[5] Arora, Sanjeev and Barak, Boaz. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st ed. , 2009.

[6] Alex Graves, Greg Wayne, Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[7] Fogel, David B. *Evolutionary computation: toward a new philosophy of machine intelligence*, vol. 1. John Wiley & Sons, 2006.

[8] Leeuwen, Jan Van and Wiedermann, Jirí. The turing machine paradigm in contemporary computing. in *Mathematics Unlimited - 2001 and Beyond. LNCS*, pp. 1139–1155. Springer-Verlag, 2000.

کتاب‌نامه

[9] Milner, Robin. Elements of interaction: Turing award lecture. *Commun. ACM*, 36(1):78–89, January 1993.

[10] Wegner, Peter. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40, 5 1997.

[11] Wegner, Peter. Interactive foundations of computing. *Theoretical Computer Science*, 192, 1998.

[12] Karp, Richard M. and Lipton, Richard J. Some connections between nonuniform and uniform complexity classes. in *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pp. 302–309, New York, NY, USA, 1980. ACM.

[13] Uhr, Leonard and Vossler, Charles. A pattern recognition program that generates, evaluates, and adjusts its own operators. in *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), pp. 555–569, New York, NY, USA, 1961. ACM.

[14] Cireşan, Dan Claudiu, Meier, Ueli, Gambardella, Luca Maria, and Schmidhuber, Jürgen. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, dec 2010.

[15] McCulloch, W.S. & Pitts, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[16] Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

[17] Cho, Sung-Bae. Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *IEEE Transactions on Neural Networks*, 8(1):43–53, 1997.

[18] Tebelskis, Joe. *Speech recognition using neural networks*. Ph.D. thesis, Siemens AG, 1995.

[19] Craven, Mark W and Shavlik, Jude W. Using neural networks for data mining. *Future generation computer systems*, 13(2):211–229, 1997.

[20] Mizuno, Hirotaka, Kosaka, Michitaka, Yajima, Hiroshi, and Komoda, Norihisa. Application of neural network to technical analysis of stock market prediction. *Studies in Informatic and control*, 7(3):111–120, 1998.

[21] Siegelmann, Hava T. and Sontag, Eduardo D. On the computational power of neural nets. in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pp. 440–449, New York, NY, USA, 1992. ACM.

[22] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[23] Hopfield, J J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[24] Cannon, Walter B. The wisdom of the body. *International Journal of Ethics*, 43(2):234–235, 1933.

[25] Belew, Richard K. and Booker, Lashon B., eds. . *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[26] Holland, John H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[27] Atmar, WIRT. Natural processes which accelerate the evolutionary search. in *Signals, Systems and Computers, 1990 Conference Record Twenty-Fourth Asilomar Conference on*, vol. 2, p. 1030. IEEE, 1990.

[28] Fisher, Ronald Aylmer. *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press, 1930.

[29] Rechenberg, I. Optimierung technischer systeme nach prinzipien der biologischen information, 1973.

کتاب‌نامه

[30] Bäck, Thomas, Hoffmeister, Frank, and Schwefel, Hans—Paul. A survey of evolution strategies. in *Proceedings of the 4th international conference on genetic algorithms*, pp. 2–9, 1991.

[31] Beyer, Hans-Georg. Toward a theory of evolution strategies: On the benefits of sex—the $(\mu/\mu, \lambda)$ theory. *Evolutionary Computation*, 3(1):81–111, 1995.

[32] Togelius, Julian and Schmidhuber, Jürgen. An experiment in automatic game design. in *CIG*, pp. 111–118, 2008.

واژه‌نامهٔ فارسی به انگلیسی

الف

ب

پ

ت

واژه‌نامهٔ فارسی به انگلیسی

واژه‌نامهٔ فارسی به انگلیسی

واژه‌نامهٔ فارسی به انگلیسی

واژه‌نامهٔ انگلیسی به فارسی

Advice	راهنما
Attention	توجه
Backpropagation	پس انتشاری
Coding	کدگذاری
Commutative Unitary Ring	حلقهٔ یکدار جابجایی
Complexity	پیچیدگی
Computable	محاسبه‌پذیر
Confguration	پیکربندی
Content-based Addressing	آدرس‌دهی بر اساس محتوا
Convergence	همگرایی
Convolutional Shift	جابجایی چرخشی
Endomorphism	درون‌ریخت
Evolutionary algorithms	الگوریتم‌های تکاملی
Feedforward Neural network	شبکهٔ عصبی پیش‌خوراند
Field	میدان
Gene	ژن
Generation	نسل
Genetic Algorithm	الگوریتم ژنتیک
Gradient Descent	یشترین کاهش
Halting Problem	مسئلهٔ توقف
Halting Set	مجموعهٔ کار
Head	اشاره‌گر

واژه‌نامهٔ انگلیسی به فارسی

مدل پنهان مارکوف .... **Hidden Markov Model**

وابستگی زمانی .... **History-dependent**

تعامل .... **Interaction**

جریان تعامل .... **Interaction Stream**

برچسب .... **Label**

حساب لاندا .... **Lambda Calculus**

آدرس‌دهی بر اساس موقعیت .... **Location-based Addressing**

یادگیری ماشین .... **Machine Learning**

گام کلان .... **Macro-step**

نوار حافظه .... **Memory Tape**

گام خرد .... **Micro-step**

شبکهٔ عصبی .... **Neural Network**

ماشین تورینگ عصبی .... **Neural Turing Machine**

ماشین غیر قطعی .... **Nondeterministic Machine**

پیشگو .... **Oracle**

والد .... **Parent**

بازشناسی الگو .... **Pattern Recognition**

چند جمله‌ای .... **Polynomial**

مسئلهٔ چاپ .... **Printing Problem**

ماشین احتمالاتی .... **Probabilistic Machine**

تقریبا احتمالا درست .... **Probably Approximately Correct**

شبکه‌های عصبی بازگردنده .... **Recurrent neural network**

مجموعهٔ بازگشتی .... **Recursive**

توابع بازگشتی .... **Recursive Functions**

مجموعهٔ شمارشی بازگشتی .... **Recursively Enumerable**

تحویل .... **Reduction**

یادگیری تقویتی .... **Reinforcement Learning**

واژه‌نامهٔ انگلیسی به فارسی

قالب . . . . **Schema Processing**

محاسبات علمی . . . . **Scientific Computing**

یادگیری نیمه‌نظارتی . . . . **Semisupervised Learning**

متمرکز . . . . **Sharpen**

ماشین استاندارد . . . . **Standard Machine**

یادگیری با نظارت . . . . **Supervised Learning**

تابع گذر . . . . **Transition Function**

تورینگ کامل . . . . **Turing Complete**

ماشین تورینگ . . . . **Turing Machine**

ماشین جهانی . . . . **Universal Machine**

یادگیری‌بدون نظارت . . . . **Unsupervised Learning**

واژه‌نامهٔ انگلیسی به فارسی

پیوست آ
سلسله مراتب یادگیری

آ.۱ یادگیری سطح صفر

یک ماشین در سطح صفرم یادگیری، دقیقا مشابه یک ماشین محاسبه‌گر معمولی است. این سطح از ماشین‌ها صرفا به عنوان مبنای بررسی یادگیری ماشین قرار دارند و در این مقاله در مورد تفاوت‌هایی که ممکن است این سطح از ماشین‌های در سطوح بالاتر ایجاد کند صحبتی به میان نیامده است.

آ.۲ یادگیری در سطوح بالاتر

یک ماشین هوشمند از سطح p به صورت بازگشتی به صورت چندگانه منظم $< \Gamma, \Sigma, \eta, \delta, \iota >$ از ماشینی با هوشمندی در یک مرحله پایین‌تر تعریف می‌گردد. این ماشین به عبارتی روشی را توصیف می‌کند که از میان چندین ماشین سطح $p - ۱$، یک ماشین خاص برگزیده شود. هر پیکربندی در یک ماشین هوشمند تشکیل شده از یک نوار حافظه و زمان محاسبه است. نوار حافظه عبارتی با طول دلخواه از الفبای $\Sigma \cup \Gamma$ است که در برگیرندهٔ اطلاعات کاری ماشین، و دقیقا یکی از ماشین‌های تحت بررسی است. زمان محاسبه نیز، عددی صحیح و مثبت است که با شروع کار ماشین مقداری برابر با صفر دارد و با هر مرحله انجام عملیات توسط ماشین مقدار آن یک واحد افزایش می‌یابد. یک مایشن هوشمند همچین در تعریف خود شامل ۳ تابع است که هر کدام قابل بیان به صورت یک ماشین از درجهٔ $p - ۱$ هستند.

- $\eta : \Gamma^* \rightarrow ((\Sigma \cup \Gamma)^*, ۰)$: این تابع که به آن تابع آماده‌سازی می‌گوییم ورودی مسئله را گرفته و اولین پیکربندی ماشین را تشکیل می‌دهد. این پیکربندی می‌تواند از لحاظ ظاهری بسیار متفاوت از ورودی مسئله باشد و یا حتی در مواردی خاص مستقل از آن باشد.
- $\iota : ((\Sigma \cup \Gamma)^*, \mathcal{N}) \rightarrow \Gamma^*$: ماشین هوشمند در هر مرحله از عملکرد خود، یک ماشین هوشمند از مرتبهٔ پایین‌تر را اجرا می‌کند. تابع ι یا تابع ورودی با توجه به پیکربندی فعلی ماشین مشخص می‌کند ورودی ماشین چه باید باشد؟
- $\delta : ((\Sigma \cup \Gamma)^*, \mathcal{N}, \Sigma^*) -> ((\Sigma \cup \Gamma)^*, \mathcal{N})$: پس اتمام محاسبهٔ ماشین سطح پایین‌تر تابع δ و یا تابع گذر، با توجه به پیکربندی فعلی و خروجی ماشین محاسبه شده، پیکربندی بعدی را مشخص می‌کند.

این دستگاه در مجموع رفتار تابعی مانند $f : \Gamma^* \rightarrow \Sigma^*$ را شبیه‌سازی می‌کند. ورودی این دستگاه در حقیقت مجموعه‌ی از اطلاعات مورد نیاز دستگاه برای شروع به کار یادگیری است و خروجی دستگاه نیز ماشینی از مرتبهٔ $p - ۱$ است. همچنین لازم به ذکر است که یک ماشین از مرتبهٔ p می‌تواند هر ماشینی از مرتبهٔ $p - ۱$ را با هر ورودی‌ای در زمان ۰ شبیه‌سازی کند، به شرط آنکه ماشین شبیه‌سازی شده، با ورودی داده شده در زمان متناهی متوقف شود.
روش کار یک ماشین هوشمند به صورت زیر است:

۱ ماشین با گرفتن ورودی و اجرای تابع η بر روی آن آغاز به کار می‌کند.

۲ سپس در هر مرحله از محاسبه به ترتیب ۳ بخش را انجام می‌دهد.

- ابتدا ماشین $m \in \Sigma^*$ را از الفبای خود استخراج می‌کند. این مرحله به این معنی است که تمام حروفی از الفبای Σ را که در پیکربندی وجود دارند را به همان ترتیب استخراج می‌شود.
- در مرحلهٔ دوم تابع ι را بر روی پیکربندی اجرا می‌کند و خروجی آن را به عنوان ورودی به ماشین m می‌دهد.
- در نهایت تابع گذر را با ورودی‌هایی شامل پیکربندی فعلی و خروجی ماشین m اجرا می‌کند و پیکربندی جدید را تشکیل می‌دهد. به عبارت دیگر

$$new\ configuration := delta(configuration, \lfloor m \rfloor(\iota(configuration))$$

در این مرحله اگر خروجی تابع گذر برابر ε بود، ماشین متوقف می‌شود و عبارت m را به عنوان خروجی باز می‌گرداند.

لازم به ذکر است که تمامی مراحل بالا در مجموع به میزان یک واحد زمانی از یک ماشین هوشمند وقت صرف می‌نمایند. این بدین معنی است که تنها تعداد دفعاتی که پیکربندی ماشین تغییر کرده است به عنوان زمان محاسبه در نظر گرفته می‌شود.

آ.۳ چند نمونه

روش‌های یادگیری که امروزه استفاده می‌شود، قریب به اتفاق به وضوح در یکی از دسته‌های این ماشین‌ها می‌گنجند. هر طبق آنچه بعدا اثبات خواهد شد این خانواده‌های از ماشین‌ها از نظر توان محاسباتی تفاوتی با یکدیگر ندارند، ولی از نظر پیچیدگی و زمان محاسبه می‌توان آن‌ها را به سادگی تفکیک نمود. این بدین معنی است که یک الگوریتم یادگیری می‌تواند ذاتا الگوریتمی از مرتبهٔ یک یادگیری باشد، حال آنکه بتوان آن را با ماشینی در مرتبهٔ صفر نیز شبیه سازی نمود.

الگوریتم‌های یادگیری به طور معمول از سه تابع تشکیل شده‌اند. این سه تابع بیانگر روشی برای مقدار دهی اولیه تابع، روشی برای بررسی وضعیت موجود و در نهایت روشی برای بهبود وضعیت موجود هستند. الگوریتم‌ها پس از مقدار دهی اولیه، در یک چرخه اقدام به بررسی و بهبود وضعیت حال خود می‌کنند و این عمل را تا زمانی ادامه می‌دهند که وضعیت موجود از یک محدودهٔ معنی بهینه‌تر گردد و یا زمان محاسبه بیش از مقدار از پیش تعیین شده‌ای شود.

در این بخش به عنوان نمونه چند الگوریتم یادگیری را بررسی می‌کنیم که چگونه و در چه سطحی می‌توان آن‌ها را در سلسله مراتب ماشین‌های هوشمند طبقه‌بندی نمود.

آ.۳.۱ شبکهٔ عصبی پیش‌خوراند

شبکه‌های عصبی مصنوعی[۱] یکی از مجموعه‌ای از شناخته شده‌ترین روش‌های یادگیری ماشین و هوش مصنوعی محسوب می‌گردند. شبکهٔ عصبی پیش‌خوراند از جمله اولین پیاده‌سازی‌های شبکه‌های عصبی بودند که اولین بار در دهه ۶۰ میلادی معرفی شدند. مبنای کار این گونه شبکه‌های عصبی ساختار نسبتا ساده‌ای دارد. یک شبکه از چندین لایه نورون تشکیل شده است که در هر لایه تمام نورون‌ها از یکدیگر مستقل‌اند ولی میان هر دو نورون در دو لایهٔ متوالی ارتباطی وجود دارد. اولین لایه از نورون‌ها متناظر با ورودی‌های شبکه است و آخرین لایه نیز بیانگر خروجی محاسبه شدهٔ شبکه است. در یک شبکه‌عصبی چند لایه، تلاش بر این است که با تغییر ضرایب اتصال میان نورون‌ها شبکه بر اساس نمونه‌های ورودی یاد بگیرد که خروجی‌های صحیح را پیش‌بینی نماید.

یادگیری یک شبکهٔ عصبی پیش‌خوراند را می‌توان به سه مرحله تفکیک نمود. مرحلهٔ نخست مقداردهی اولیه شبکه است که به طور معمول به هر کدام از اتصالات میان نورون‌ها ضریبی تصادفی نسبت داده می‌شود. سپس شبکه برای یکی از نمونه‌ها شبکه‌ای که با مقداردهی اولیه شده خروجی را پیش‌بینی می‌کند. این خروجی به احتمال زیاد خروجی مورد انتظار ما نیست در نتیجه در مرحلهٔ سوم با توجه به خروجی مورد انتظار مقادیر اتصالات را تغییر می‌دهیم که در مورد آن ورودی خاص جواب مناسب‌تری حاصل شود. در نهایت مراحل دوم و سوم را برای نمونه‌ها دیگر تکرار می‌کنیم. این عملیات تا زمانی ادامه خواهد داشت که شبکه خروجی نسبتا مناسبی را به ازای تمام ورودی‌ها بتواند پیش‌بینی کند.

آنچه از شبکه‌های عصبی چند لایه بیان شد نشان می‌دهد که اجزای یک شبکه تناظر مستقیم با اجزای معرفی کنندهٔ یک ماشین هوشمند سطح یک دارند. نمونه‌های ورودی که ماشین باید بر اساس آنها آموزش ببیند عباراتی هستند که در زبان Γ بیان می‌شوند، محاسبهٔ خروجی یک شبکه نیز عبارتی است که توسط الفبای Σ بیان شده است و مقادیر ارتباطات میان نورون‌ها در این ماشین به صورت ذاتی وجود دارند. تابع ورودی یک ماشین، در مدل یادگیری یک شبکهٔ عصبی معادل تابعی است که در هر بار یکی از نمونه‌ها را انتخاب می‌کند که در این مرحله قرار است ماشین آموزش ببیند. مرحلهٔ پس انتشار شبکه نیز توسط تابع δ مدل می‌شود.

با توصیفات فوق می‌توان به آسانی بررسی نمود که زمان اجرای ماشین هوشمند دقیقا برابر تعداد مراحلی است که برای یادگیری یک شبکهٔ عصبی پیش‌خوراند صرف می‌گردد.

[۱] Artificial neural network

آ.۲.۳ مدل پنهان مارکوف

محاسبهٔ نتیجهٔ حاصل از یک زنجیرهٔ مارکوف از جمله کارهایی است که یک ماشین با هوشمندی سطح صفر به آسانی می‌تواند انجام دهد. آموزش توسط «مدل پنهان مارکوف»[۲] ولی، مسئله از مرتبهٔ ماشین‌های با یادگیری مرتبهٔ اول است. به عنوان مثال در این نگاره روش forward-backward را بررسی می‌نماییم. این شیوه از تکرار متوالی دو الگوریتم تشکیل شده، که در مرحلهٔ forward وضعیت فعلی مدل بررسی می‌گردد و در مرحله backward مدل با توجه به اشتباهات احتمالی ظاهر شده در مرحلهٔ قبل تصحیح می‌گردد.

- η تابع آماده‌سازی در یک ماشین آموزشگر کافی است در کنار ورودی مسئله، یک پیاده‌سازی از الگوریتم forward که در زبان Σ کدگذاری شده است را اضافه نماید. تمام پارامترهای موجود در این پیاده‌سازی می‌تواند مقادیری تصادفی داشته باشند.
- ι در این ماشین تابع انتخاب به سادگی تمام ورودی مسئله را به ماشین سطح پایین‌تر تحویل می‌دهد.
- δ تابع گذر نیز بر اساس خروجی‌های الگوریتم forward ضرایب مدل را تغییر می‌دهد تا احتمال رخداد ورودی را بیشینه نماید.

همانطور که مشاهده شد، کلیات مراحل کار این ماشین، مشابه مراحل کار یک آموزشگر شبکهٔ عصبی چند لایه است، و تفاوت اصلی میان این دو الگوریتم در جزئیات رفتار آنها در سطح صفرم یادگیری است.

آ.۳.۳ الگوریتم ژنتیک

«الگوریتم ژنتیک»[۳] اختلافی بنیادی با الگوریتم‌های قبلی بررسی شده دارد. در الگوریتم ژنتیک بر خلاف آموزشگرهای قبلی که یک نسخه از برنامهٔ نهایی را از ابتدا به صورت تصادفی تولید می‌نمودند و در ادامه به بهبود آن می‌پرداختند، چندین نسخهٔ مختلف از برنامه‌های حل مسئله به صورت تصادفی تولید می‌شوند و این مجموعه به صورت همزمان در هر مرحله بهبود می‌یابد. به عبارت دیگر الگوریتم آموزش به جای آموزش تنها یک نسخه از ماشین پایانی وظیفهٔ آموزش چندین نسخه را به صورت همزمان به عهده دارد. همین تفاوت باعث می‌گردد که الگوریتم ژنتیک به صورت ذاتی از یادگیری مرتبهٔ دوم برخوردار باشد. در ادامه توضیح می‌دهیم هر کدام از سطوح یادگیری در یک ماشین متناظر با الگوریتم ژنتیک چه وظیفه‌ای به عهده دارند:

- مرحلهٔ صفرم یادگیری صرفا یک ماشین حل مسئله و بیانگر یکی از گونه‌ها در الگوریتم ژنتیک است. این ماشین وظیفهٔ محاسبهٔ خروجی مناسب بر اساس ورودی‌های مسئله را بر عهده دارد.
- در سطح اول یادگیری ماشینی قرار می‌گیرد که وظیفهٔ انتخاب بهترین گونه را از بین مجموعه‌ای مشخص از گونه‌ها بر عهده دارد. مجموعهٔ گونه‌ها جزو توصیف این ماشین هستند و مستقل از ورودی ماشین می‌باشند.
- در نهایت الگوریتم ژنتیک ماشینی از سطح دوم یادگیری خواهد بود. در این ماشین هر حالت بیانگر مجموعه‌ای از گونه‌ها (و یا همان ماشین‌های سطح یک) است. در هر گذر این ماشین به بررسی گونه‌های موجود در حالت فعلی خودش می‌پردازد و تعدادی از آنها را جهش می‌دهد یا ادغام می‌کند. همچنین اگر بهترین گونهٔ موجود در مجموعه جوابی قابل قبول تولید می‌نمود آن جواب پردازش ماشین خاتمه می‌یابد.

چند نکته در مورد توصیف بالا قابل تامل است:

- زمان اجرای ماشن بالا برابر تعداد نسل‌هایی است که الگوریتم ژنتیک برای تولید جواب بهینه پیموده است.

[۲] Hidden Markov Model
[۳] Genetic Algorithm

- خروجی اولیه ماشین مرتبهٔ دوم بالا شامل تمام گونه‌هایی است که در آخرین نسل وجود داشته است، و در نتیجه برای یافتن بهترین گونه باید ماشین مرتبهٔ اول حاصل از اجرای الگوریتم ژنتیک را به صورت جداگانه اجرا نمود.

A Model for Analysis of computational learning

Abstract

The subject of this thesis is about a certain set of algorithms that try to indirectly solve problems. Instead of a programmer crafting an algorithm to solve a problem, these algorithms learn a solution themselves. These methods are usually studied in the Probably Accurately Correct (PAC) learning model. Although, PAC Learning is a generally accepted model, it falls short to describe certain aspects of learning algorithms. Many learning methods rely on convergence to minimize error, or maximize their fitness, yet the PAC model doesn't explicitly provide any means to measure these behaviours. In this thesis, we first go through different models related to computational requirements of learning and convergence. We then continue by describing PAC model, and finally proceeded by introducing a new model along with metrics to examine convergence properties of a learning algorithm. We continue by showing that these new metrics doesn't undermine the generality of PAC model, and a problem is PAC learnable if and only if it's gradually learnable. This thesis doesn't provide any actual analysis of existing algorithms, and leaves it for future researchers interested in this field.

Keywords: *Machine Learning, Computabilty Theory, Computation model, Turing machine*

Sharif University of Technology
Department of Mathematics

A thesis Submitted in partial fulfillment of the
requirements for the M.Sc. degree

A Model for Analysis of computational learning

By

Ali Sattari Javid

Supervisor

Professor Daneshgar

Jan 2017