حافظه زمانی سلسله‌مراتبی

HIERARCHICAL TEMPORAL MEMORY

زمستان ۹۳ / January 2015

نگارش: مسعود عباسیان
abasian@aut.ac.ir

چکیده

بسیاری از فعالیت‌ها هستند که انسان به راحتی انجام می‌دهد درصورتی‌که کامپیوترها همچنان از انجام آن‌ها عاجز می‌باشند. فعالیت‌هایی همچون شناسایی الگوهای تصویری، شناسایی زبان گفتاری، شناسایی اشیا با استفاده از قدرت لامسه و هدایت اجسام در محیط‌های پیچیده، برای انسان به‌صورت ساده‌ای قابل انجام هستند. بااین‌حال برخلاف تحقیقات دهه‌های اخیر، تعداد کمی الگوریتم‌های مناسب برای به رسیدن به عملکرد فعالیت‌های انسانی در کامپیوتر به دست آمده است.

فعالیت‌هایی که در بالا به آن اشاره شد، توانایی‌هایی هستند که به‌صورت کلی توسط قشر مغزی[1] (لایه غشایی مغزی حاوی نورون‌ها یا قشر مخ) انجام می‌شوند. حافظه زمانی سلسله‌مراتبی[2] یا به‌اختصار HTM یک تکنولوژی است که سعی در مدل کردن چنین فعالیت‌هایی را دارد. این حافظه برای ساخت ماشین‌هایی استفاده می‌شود که هدف آن‌ها در انجام چنین فعالیت‌های ادراکی است که توسط انسان و به‌سادگی انجام می‌شود.

در این گزارش تعاریف و عملکرد حافظه زمانی سلسله‌مراتبی را شرح می‌دهیم. فصل اول یک بازبینی بر این حافظه، مختصری از اهمیت سازمان‌های سلسله‌مراتبی، بازنمایی توزیع‌های تنک و مبحث یادگیری تغییر حالت بر اساس زمان است. فصل دوم الگوریتم یادگیری قشر مغزی حافظه زمانی سلسله‌مراتبی را به تفضیل شرح می‌دهیم. فصل سوم و چهارم به تشریح الگوریتم یادگیری حافظه زمانی سلسله‌مراتبی در دو بخش می‌پردازیم که به ترتیب اصطلاحاً متمرکز کننده مکانی و دیگری متمرکز کننده زمانی این الگوریتم نامیده می‌شوند. بعد از فصل‌های ۳ تا ۵، مهندسین نرم‌افزار باتجربه قادر خواهند بود بر اساس توضیحات داده‌شده و شبه کد معرفی‌شده، این الگوریتم را پیاده‌سازی و تولید کنند.

کلیدواژه: قشر مغزی، حافظه زمانی سلسله مراتبی، توزیع تنک، قشر مخ، متمرکز کننده زمانی، متمرکز کننده مکانی

[1] neocortex

[2] Hierarchical Temporal Memory

فهرست مطالب

فهرست شکل‌ها

۱. مقدمه

یکی از پر چالش‌ترین مباحثی که در یادگیری ماشین و هوش مصنوعی مطرح می‌شود، بحث الگوریتم‌های یادگیری هستند که الگوهای متغیر را از محیط اطراف بگیرند و بر اساس این الگوها که در طول زمان تغییر می‌کنند، یادگیری را انجام دهند. از جمله این مسائل می‌توان به شناسایی الگو[۱]، کنترل[۲]، دقت[۳] و آموزش افزایشی[۴] اشاره کرد. کاربردهای خاصی نیز مانند شناسایی زبان اشاره[۵] (به‌اختصار SLR) وجود دارد که در این گزارش به عنوان یک کاربرد از آن یاد شده است [1]. برای این‌گونه مسائل مدل‌هایی تاکنون ارائه شده است که از جمله آن می‌توان به روش‌های کلی مانند الگوریتم‌های منطق فازی[۶]، انتقال موجک[۷]، شبکه‌های عصبی مصنوعی[۸]، کلاس بندهای الهام گرفته از بیوانفورماتیک[۹] و یا مدل مخفی مارکوف[۱۰] اشاره کرد [1].

اما این روش‌ها دارای مشکلاتی هستند که باعث می‌شود در بعضی از مسائل، کارایی لازم را نداشته باشند. این مسائل، مواردی هستند که الگوهای ورودی و یادگیری ماشین، بر حسب زمان در حال تغییر است و یک خاصیت پویا در هر لحظه در سیستم وجود دارد [4] یا مسائلی که بحث شناسایی سری‌های زمانی چند متغیره را دارند. شبکه حافظه زمانی سلسله مراتبی این قابلیت را دارد که بر چنین مسائلی غلبه کند. این شبکه بر اساس پایه‌های بیز بنانهاده شده است [1] و تا حد امکان سعی در پیاده‌سازی قشر نورونی مغز انسان را دارد [3]. دو اصل مهم در این شبکه (حافظه) روال سازمان‌دهی سلول‌ها و ساختار آن و همچنین رمزنگاری اطلاعات ورودی است[1] که با بقیه شبکه‌های عصبی سنتی تفاوت دارد[3].درنهایت اینکه این شبکه قادر است یادگیری و استنتاج را در طول زمان و برای ورودی‌های متغیر و دنباله‌دار، به نحو خوبی انجام دهد [1].

تاریخچه

تئوری حافظه زمانی سلسله‌مراتبی در سال ۲۰۰۴ در کتاب on intelligence، در گزارش‌های منتشرشده توسط شرکت نومنتا[۱۱] و همچنین در مقاله‌های بازبینی‌شده به وسیله کارکنان این شرکت تفسیر شده است. در این گزارش جدیدترین نسخه این الگوریتم که به نام زتا۱[۱۲] نام‌گذاری شده است را معرفی می‌کنیم. برای مدت کوتاهی این الگوریتم با نام بازنمایی توزیع‌شده چگالی ثابت[۱۳] شناخته می‌شد یا به‌اختصار FDR. ولی دیگر از این نام آن

---

[۱] Pattern Recognition

[۲] Control

[۳] attention

[۴] Incremental Learning

[۵] sign language recognition

[۶] fuzzy logic

[۷] wavelet transform

[۸] artificial neural networks

[۹] bioinspired classifiers

[۱۰] Hidden Markov Models

[۱۱] Numenta

[۱۲] Zeta 1

[۱۳] Fixed-density Distributed Representations

استفاده نمی‌شود. هم‌اکنون جدیدترین الگوریتم را، الگوریتم یادگیری قشری حافظه زمانی سلسله‌مراتبی[۱] یا فقط الگوریتم یادگیری حافظه زمانی سلسله‌مراتبی میگویند [2].

کتاب on intelligence توسط مؤسس شرکت نومنتا جف هاوکینز[۲] با همکاری ساندرا بلکسلی[۳] نوشته شده است [3] که در آن به حافظه زمانی سلسله‌مراتبی با این لفظ اشاره نشده و تنها به‌صورت ساده و غیرتخصصی، تئوری و علوم نهفته در پشت این حافظه را شرح داده است [2].

شرکت نومنتا

این شرکت در سال ۲۰۰۵ برای پیشبرد تکنولوژی حافظه زمانی سلسله‌مراتبی برای دو مقوله تجاری‌سازی و علمی تشکیل شد. به همین منظور این شرکت به‌صورت کامل در حال ساخت اسناد بر اساس پیشرفت‌ها و کشفیاتی است که تا به حالا داشته‌اند؛ و همچنین الگوریتم‌های تولیدشده توسط این شرکت به‌گونه‌ای است که در هر دو زمینه تجاری و علمی بتوان از آن بهره برد که استفاده از نرم‌افزارهای این شرکت و همچنین مالکیت معنوی برای مقاصد علمی آزاد است. درحالی‌که استفاده از این نرم‌افزارها برای مقاصد تجاری نیاز به کسب اجازه از این شرکت دارد. مقر این شرکت در کالیفرنیا[۴] در شهر ردوود سیتی[۵] است که به‌صورت خصوصی تأسیس گردیده است [2].

۲. مروری بر حافظه زمانی سلسله‌مراتبی

حافظه زمانی سلسله‌مراتبی یک مکانیزم یادگیری ماشین است که هدف آن دسترسی به ساختار و الگوریتمی با خواص قشر مخ است. قشر مخ یک مرکزی از هوش در مغز پستانداران است. بینایی، شنوایی، لامسه، حرکات، زبان و برنامه‌ریزی تماماً توسط این قسمت ایفا می‌شود. فرض کنید یک سری مجموعه متنوع از کاربردهای ادراکی داریم و شما انتظار دارید که قشر مخ یک مجموعه همسان از الگوریتم‌های خاص عصبی را پیاده‌سازی کند. این یک اتفاق و حالت خاص نیست. قشر مخ یک الگوی یکنواخت قابل‌توجه از مدارات عصبی را نمایش می‌دهد. مشاهدات بیولوژیکی اظهار دارند که قشر مخ یک مجموعه عمومی از الگوریتم‌ها را برای اجرای توابع مختلف هوشی را پیاده‌سازی می‌کنند و به این صورت نیست که برای هر کاربرد خاص، یک الگوریتم و الگوی خاصی را به کار بندد [3].

حافظه زمانی سلسله‌مراتبی به‌عنوان یک چهارچوبی برای آشنایی و فهمیدن ساختار قشر مخ و توانایی‌های آن است. در حال حاضر این حافظه به‌گونه‌ای پیاده شده است که تا حد کافی، زیرمجموعه‌ای از این توانایی‌های مغزی را پیاده کند که ارزش تجاری و علمی دارند [2].

برنامه‌نویسی برای این مدل شبکه (حافظه) با برنامه‌ریزی سنتی تفاوت دارد. برنامه‌های امروزی به این صورت می‌باشند که یک برنامه خاص را برای یک مسئله خاص می‌نویسند؛ اما عملکرد این حافظه به این صورت است که با قرار گرفتن در راستای یک سری از حس‌گرهای اطلاعاتی، آموزش داده می‌شود. توانایی‌های آن به این صورت آشکار می‌شود و عمل می‌کند که ما این الگوریتم را در معرض چه نوع اطلاعات و حس‌گرهایی قرار داده‌ایم؛ یعنی عملکرد آن بر حسب حس‌گرهای اطلاعاتی و داده‌های ورودی‌هایی، متفاوت خواهد بود [2].

---

[۱] HTM Cortical Learning Algorithms

[۲] Jeff Hawkins

[۳] Sandra Blakeslee

[۴] California

[۵] Redwood City

حافظه زمانی سلسله‌مراتبی را می‌توان یک نوع شبکه عصبی دید چون هر ساختاری که سیستم عصبی مغز را پیاده کند به‌عنوان شبکه عصبی شناخته می‌شود. این شبکه نورون‌هایی را مدل می‌کند که در سطرها، لایه‌ها، ناحیه‌ها و در سلسله‌مراتب مختلف قرار دارند. این حافظه یک نوع شبکه عصبی مدرن است که تمام جزییات بالا برای آن لحاظ شده است و از این مقوله از سایر شبکه‌های سنتی عصبی جدا می‌شود [3].

همان‌گونه که از نام حافظه زمانی سلسله‌مراتبی مشخص است، این ساختار یک حافظه محسوب می‌شود که سعی در ذخیره‌سازی الگوهای زیادی از زمان و دنباله‌ها را دارد. روشی که برای ذخیره‌سازی اطلاعات در این شبکه استفاده می‌شود به‌صورت منطقی از سایر شبکه‌های استاندارد و سایر برنامه‌های امروزی جدا می‌شود. برنامه‌های امروزی به‌صورت مسطح است و تصویری ذاتی از زمان را در بر ندارند. ساختار حافظه‌ای امروزی به‌صورت مسطح می‌باشند و یک برنامه‌نویس می‌تواند هر مدل از سازمان‌های اطلاعاتی و ساختاری را روی این حافظه مسطح کامپیوتری پیاده‌سازی کند. آن‌ها می‌توانند چگونگی و محل قرار گرفتن اطلاعات را کنترل کنند. برخلاف مواردی که گفته شد، ساختار حافظه زمانی سلسله‌مراتبی محدودکننده است زیرا به‌صورت سلسله‌مراتبی و بر پایه زمان بنا شده است. اطلاعات در این شبکه به‌صورت توزیع‌شده[1] ذخیره می‌شوند به‌نحوی‌که نمی‌توان گفت که مقدار خاصی از مسئله در متغیر و یا محل خاصی از این حافظه وجود دارد درنتیجه نحوه و محل قرار گرفتن اطلاعات از دست برنامه‌نویس خارج است. برنامه‌نویس فقط می‌تواند سایز ساختار سلسله‌مراتبی و نحوه آموزش سیستم را مشخص کند و اینکه اطلاعات به چه صورت و در کجا ذخیره می‌شوند بر عهده حافظه زمانی سلسله‌مراتبی است.

هرچند سیستم ذخیره اطلاعات در این حافظه متفاوت از ذخیره‌سازی‌های کلاسیک است ولی همچنان می‌توان از اهداف کلی ذخیره‌سازی در کامپیوترهای امروزی به‌منظور استفاده از حافظه زمانی سلسله‌مراتبی استفاده کرد و ویژگی‌های کلیدی همچون سلسله‌مراتبی بودن، زمان و بازنمایی توزیع‌شده را اجرا کرد.

در این گزارش سعی شده است که این شبکه را با استفاده از مثال‌های واقعی که مربوط به فعالیت‌های انسانی است شرح بدهیم. هرچند باید در نظر داشت که حافظه زمانی سلسله‌مراتبی توانایی‌های کلی و جامعی دارد و مختص فعالیت‌های خاص‌منظوره نیست. در پیاده‌سازی‌هایی که از این شبکه شده است، فعلاً قادر بوده‌اند که حس‌گرهای غیرانسانی مانند حس‌گرهای با اشعه‌ی مادون‌قرمز و یا رادارها را با شبکه ادغام کنند و یا در حالت دیگری داده‌های خالصی همچون اطلاعات مالی فروشگاه‌ها، اطلاعات هواشناسی، الگوهای ترافیک اینترنت و یا متن را به آن بدهند. حافظه زمانی سلسله‌مراتبی یک ماشین یادگیری و تخمین زننده هست که می‌تواند تعداد زیادی از مسائل با انواع داده‌ها را برای ما حل کند [2].

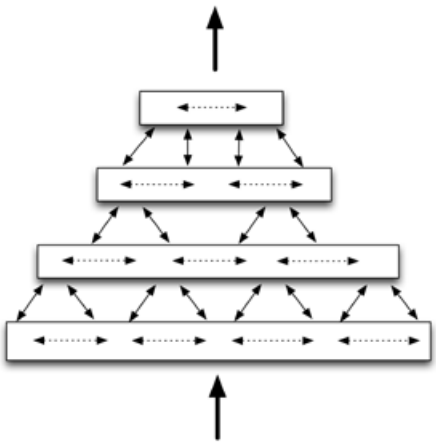
۱.۲. قاعده‌های کلی حافظه زمانی سلسله‌مراتبی

در این بخش قاعده‌های کلی حافظه زمانی سلسله‌مراتبی را بررسی می‌کنیم. چرا ساختار سلسله‌مراتبی مهم است؟ چگونه ناحیه‌های حافظه زمانی سلسله‌مراتبی ساخته می‌شوند؟ چرا اطلاعات به‌صورت توزیع‌شده بازنمایی می‌شوند و چرا اطلاعات بر پایه زمانی بسیار ضروری هستند؟ این‌ها سوالاتی است که در این بخش می‌توانیم به آن‌ها پاسخ بدهیم.

[1] distributed

۱.۱.۲. سلسله‌مراتب

این شبکه از یک سری ناحیه‌هایی[۱] که در یک سلسله‌مراتب چیده شده است تشکیل می‌شود. این نواحی اصلی‌ترین واحد حافظه و تخمین در این شبکه به شمار می‌روند که در بخش بعدی به تفضیل در مورد آن‌ها صحبت خواهیم کرد. به‌طورکلی هر ناحیه، یک سطح[۲] از سلسله‌مراتب را نشان می‌دهد. همان‌طور که به طرف بالای شبکه می‌رویم، همیشه به سمت همگرایی پیش می‌رویم. المان‌های مختلف در یک ناحیه پایین (فرزندان) به یک المان در یک ناحیه بالا (والد) همگرا می‌شوند. از طرفی به خاطر وجود اتصالات فیدبک، اطلاعات می‌توانند در این سلسله‌مراتب به سمت پایین نیز حرکت کنند. (به‌طورکلی کلمات ناحیه و سطح در این شبکه مشابه هستند ولی در این گزارش زمانی که در مورد ساختار درونی شبکه صحبت می‌کنیم از کلمه ناحیه و هنگامی‌که در مورد نقش یک ناحیه صحبت می‌کنیم از کلمه سطح استفاده خواهیم کرد). شکل ۱-۱ ساختار ناحیه‌ای و سلسله‌مراتبی این حافظه و همچنین جهت همگرایی آن را نشان می‌دهد[2] .



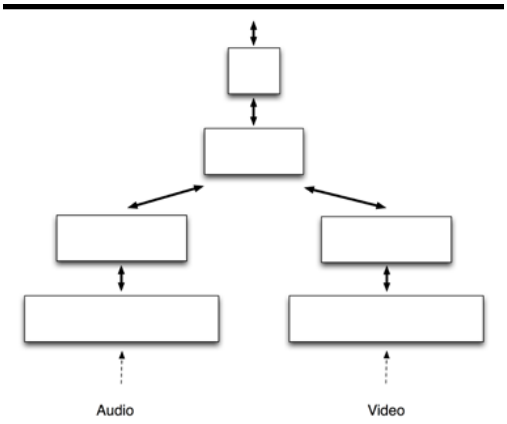
شکل ۱ - دیاگرام ساده از ۴ ناحیه حافظه زمانی سلسله‌مراتبی در ۴ سطح سلسله‌مراتبی [2]

این امکان وجود دارد که تعدادی از این شبکه‌ها را با هم ترکیب کنیم. این نوع ساختار زمانی مفید است که شما اطلاعاتی از بیش از یک منبع و یا حس‌گر را داشته باشید. به‌عنوان‌مثال یک شبکه می‌تواند اطلاعات صوتی را پردازش کند و شبکه دیگر اطلاعات تصویری را. همگرایی برای هر شبکه به‌صورت جداگانه است و انشعابات همگرایی کلی نیز، تنها به سمت بالای این شبکه خواهد بود. شکل ۱-۲ نحوه ترکیب شدن چند مدل از این حافظه و نحوه همگرایی آن را به نمایش گذاشته است [2].

مزایای ساختار سلسله‌مراتبی بسیار زیاد است. این ساختار به‌صورت کارآمدی میزان زمان آموزش و همچنین حافظه استفاده‌شده را کاهش می‌دهد. به خاطر اینکه الگوهای یاد گرفته‌شده در هر سطح از سلسله‌مراتب، دوباره استفاده می‌شوند زمانی که در یک طرح جدید در یک سطح بالاتر ترکیب می‌شوند. برای یک مثال، سیستم بینایی را در نظر می‌گیریم. در پایین‌ترین سطح از این سلسله‌مراتب، مغز شما اطلاعاتی در مورد قسمت‌های ریز و بسیار کوچک از بخش بینایی را نگه‌داری می‌کنند مانند لبه‌ها و کناره‌های تصویر. یک لبه، یک مؤلفه بنیادی از بسیاری از اشیا در جهان است.

[۱] Region

[۲] Level



شکل ۲ – ترکیب دو حافظه زمانی سلسله‌مراتبی متشکل از دو ورودی صوت و تصویر [2]

این ترکیبات سطح پایین، باهم در سطح‌های میانی ترکیب می‌شوند تا مؤلفه‌های پیچیده‌تری همچون منحنی‌ها و بافت‌های شی را نگه‌داری و پردازش کنند. به‌عنوان‌مثال منحنی‌های اطراف گوش و یا منحنی اطراف لاستیک ماشین؛ و این اطلاعات و الگوهای سطح میانی ترکیب می‌شوند و در سطح‌های بالاتر مفاهیم پیچیده‌تر و سطح بالاتری را بازنمایی می‌کنند. به‌عنوان نمونه سر انسان‌ها، ماشین‌ها، خانه‌ها و مفاهیم دیگر. برای یادگیری یک شی سطح بالای جدید، دیگر نیاز به یادگیری مجدد مؤلفه‌های آن نیست. برای واضح‌تر شدن این جمله فرض کنید زمانی که شما کلمه جدیدی را می‌شنوید دیگر نیاز نیست که دوباره حروف و سیلاب‌ها و یا آواهای آن را دوباره یاد بگیرید.

اشتراک‌گذاری این بازنمایی‌ها در یک ساختار سلسله‌مراتبی باعث می‌شود که از رفتارهای قابل پیش‌بینی تعمیمی داشته باشیم. زمانی که شما یک حیوان جدید را می‌بینید، اگر شما یک دهان و دندان از این حیوان را مشاهده کنید، مسلماً برداشت شما این است که این حیوان با دهان غذا می‌خورد و ممکن شما را گاز بگیرد. این یک مثال بسیار ساده است که ذهن انسان به راحتی آن را درک می‌کند ولی شبکه‌های عصبی به راحتی چنین چیزی را نمی‌توانند ارائه دهند. پس ساختار سلسله‌مراتبی باعث می‌شود که اشیا جدید در جهان با استفاده از ارث‌بری از خصوصیات مشخص شده و دانسته از زیر مؤلفه‌های آن به وجود بیایند.

حال سوالی پیش می‌آید که یک سطح چقدر می‌تواند یادگیری داشته باشد؟ یا به‌صورت دیگر ما در یک شبکه حافظه زمانی سلسله‌مراتبی به چند سطح نیاز داریم. در این مورد بین مقدار حافظه‌ای که به هر سطح اختصاص داده می‌شود و تعداد سطوح مورد نیاز، تناسب است. خوشبختانه حافظه زمانی سلسله‌مراتبی بهترین بازنمایی ممکن در هر سطح، بر اساس اطلاعات آماری که از ورودی‌ها و مقدار منابع اختصاص داده شده به دست آورده است را به‌صورت خودکار یاد می‌گیرد. اگر شما حافظه بیشتری را به یک سطح اختصاص دهید، آن سطح با یک بازنمایی پیچیده‌تر و اندازه‌ای بزرگ‌تر شکل می‌گیرد که این شبکه ما را به این سمت می‌برد که تعداد سطوح کمتری را لازم داشته باشیم و برعکس.

تا اینجا ما مسائل سختی را مطرح کردیم مانند استنتاج تصویری؛ اما خیلی از مسائل باارزش، خیلی ساده‌تر از بینایی هستند و یک سطح از حافظه برای آن‌ها کافی خواهد بود. برای مثال ما در یک آزمایش، یک حافظه زمانی سلسله‌مراتبی را برای پیش‌بینی اینکه یک فردی که در حال وبگردی است، کلیک بعدی مشابه آن صفحه را در کدام لینک خواهد زد امتحان کردیم که با همین یک سطح، شبکه قادر به انجام خوب این پیش‌بینی بوده است.

این مسئله شامل قرار دادن اطلاعات کلیک در اینترنت با جریان حافظه زمانی سلسله‌مراتبی است. این مسئله فضای سلسله‌مراتبی کمی دارد و یا اصلاً وجود ندارد و بیشتر نیاز به به‌دست آوردن اطلاعات زمانی دارد که با استفاده

از شناسایی الگوهای عمومی یک کاربر، پیش‌بینی کند دفعه بعد کجا را کلیک خواهد کرد. الگوریتم‌های یادگیری زمانی در حافظه زمانی سلسله‌مراتبی برای چنین مسائلی ایده آل خواهند بود.

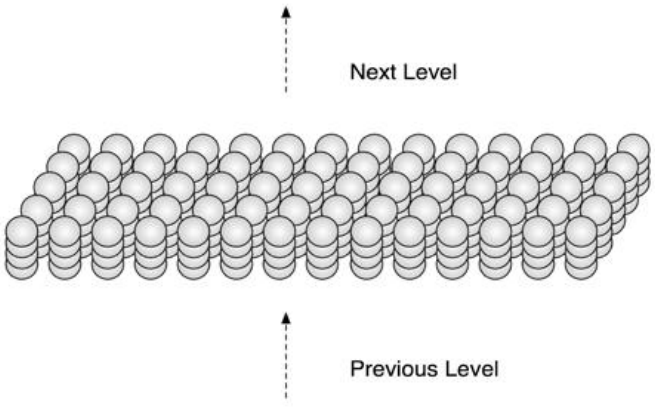
به‌طور خلاصه، سلسله‌مراتب باعث کاهش زمان آموزش، کاهش حافظه و معرفی یک ساختار تعمیم‌یافته است. هرچند بسیاری از مسائل ساده‌ی پیش‌بینی با یک حافظه زمانی سلسله‌مراتبی تک سطحی نیز می‌تواند حل شود [2].

۲.۱.۲. ناحیه‌ها

مفهوم ناحیه در یک ساختار سلسله‌مراتبی از مباحث زیست‌شناسی می‌آید. قشر مخ، یک صفحه بزرگ متشکل از بافتی نورونی است که در حدود ۲ میلی‌متر ضخامت دارد. زیست شناسان این قشر را به ناحیه‌های مختلف تقسیم کرده‌اند البته بر اساس نحوه اتصالاتی که با هم دیگر دارند. بعضی از این نواحی به‌صورت مستقیم ورودی‌ها را دریافت می‌کنند و بعضی دیگر زمانی ورودی را دریافت می‌کنند که تعدادی از ناحیه‌های دیگر، ورودی را به این ناحیه پاس داده باشند. به این مدل اتصال ناحیه به ناحیه[۱] گویند که سلسله‌مراتب را تعریف می‌کند [2].

تمام این نواحی در جزییات شبیه یکدیگر هستند. هرچند آن‌ها سایزهای مختلف و در سطح‌های مختلف قرار دارند اما به‌صورت کلی شبیه یکدیگرند. اگر یک قسمتی از این ناحیه قشری دو میلی‌متری را ببرید، شما ۶ لایه را مشاهده می‌کنید: پنج لایه از سلول‌ها و یک لایه غیر سلولی (بعضی از استثنائات وجود دارد اما این یک روال کلی در این مورد است). هر لایه در یک ناحیه تعدادی زیادی اتصالات درونی دارد. حافظه زمانی سلسله‌مراتبی هم شبیه چنین ساختاری است که یک صفحه‌ای از اتصالات زیاد سلولی است که مرتب شده‌اند. لایه ۳ قشر مخ یکی از اصلی‌ترین لایه‌های پیش‌خور[۲] از نورون‌هاست. سلول‌های داخل یک ناحیه تقریباً معادل نورون‌ها در لایه سوم در یک ناحیه از قشر مخ هستند.

در شکل ۱-۳ یک قسمت از یک ناحیه حافظه زمانی سلسله‌مراتبی نمایش داده شده است. هر ناحیه شامل تعداد زیادی سلول است؛ که این سلول‌ها، در یک آرایه دوبعدی از ستون‌ها چیده شده‌اند. این شکل یک قسمت کوچکی از یک ناحیه در حافظه را با ۴ سلول در هر ستون نشان می‌دهد. هر ستون به تعدادی مجموعه ورودی و هر سلول به سلول‌های دیگر در آن ناحیه متصل است. توجه داشته باشید که حافظه زمانی سلسله‌مراتبی - با ساختار ستونی خودش - معادل یک لایه از نورون‌ها در یک ناحیه از قشر مخ است [2].



شکل ۳- یک قسمت از یک ناحیه حافظه زمانی سلسله‌مراتبی [2]

[۱] region-to-region connectivity

[۲] feed-forward layers

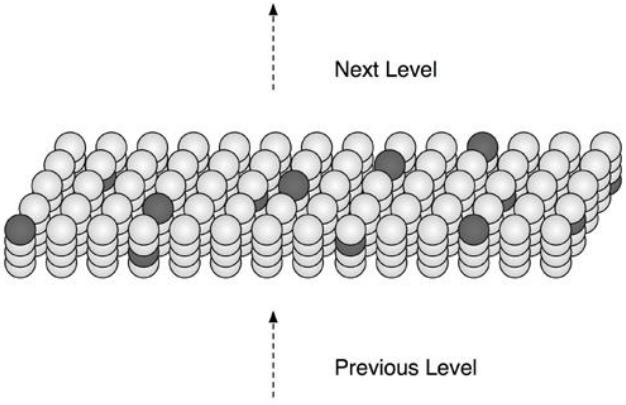
۳.۱.۲. بازنمایی توزیع‌شده تنک

بااینکه در ساختار قشر مخ، اتصالات درونی بسیار بالایی وجود دارد ولی یک سری از نورون‌ها به نام نورون‌های بازدارنده باعث می‌شوند فقط تعدادی از این نورون‌ها در هر ناحیه فعال و در یک زمان فعال باشند. پس اطلاعات در مغز، همیشه در یک سری از نورون‌های فعال ذخیره می‌شود. به این مدل رمزگذاری به‌اصطلاح بازنمایی توزیع‌شده تنک[۱] میگویند. تنک بودن به این معنی است که فقط درصد کمی ا ز نورون‌ها در یک لحظه فعال هستند. توزیع‌شده به این معنی است که فعال‌سازی خیلی از نورون‌ها نیازمند بازنمایی الگوی خاصی باشد. یک نورون فعال دلالت بر بعضی معانی را دارد اما این باید در مفهوم جمعیت نورون‌ها برای دلالت بر یک معنی کلی تفسیر شود؛ یعنی از یک نورون نمی‌توان یک الگوی ذخیره‌شده را بازیابی کرد بلکه باید همراه با نورون‌های دیگر، در یک مجموعه به رمزگشایی الگو پرداخت.

با این تفاسیر یک شبکه حافظه زمانی سلسله‌مراتبی فقط با داده‌هایی که به‌صورت پراکنده توزیع‌شده باشند کار می‌کند ولا غیر. معمولاً داده‌های ورودی به صورت بازنمایی توزیع‌شده هستند اما به‌صورت پراکنده نیستند. به همین دلیل اولین چیزی که باید در نواحی در نظر گرفت تبدیل ورودی‌ها به‌صورت بازنمایی توزیع‌شده تنک است.

برای مثال یک ناحیه ۲۰ هزار بیت ورودی دریافت می‌کند. حالات ممکنی که ورودی می‌تواند داشته باشد بر حسب اینکه هرکدام از بیت‌ها صفر یا یک باشند بسیار زیاد است و از طرفی در هرلحظه ممکن است بیت‌ها، تغییرات زیادی را تجربه کنند. یک ناحیه باید این ورودی‌ها را به یک بازنمایی داخلی با ۱۰ هزار بیت جایی که دو درصد- یا ۲۰۰ بیت - در یک لحظه فعال می‌باشند، تبدیل کند. این کار باید بر اساس اینکه چه تعداد از ورودی‌ها مقدار یک دارند انجام شود. این بازنمایی داخلی در حین اینکه ورودی واردشده به ناحیه در طول زمان تغییر کند، عوض می‌شود، اما همیشه به این صورت است که حدود ۲۰۰ بیت از ۱۰ هزار بیت فعال باشد.

این مکانیزم شاید چنین باوری را به وجود بیاورد که با محدود کردن ورودی‌ها به چنین ساختاری، احتمال از دست دادن اطلاعاتی را خواهیم داشت جایی که تعداد ورودی‌های ممکن در الگوی ورودی بسیار بیشتر از بازنمایی ممکن در یک ناحیه باشد. بااین‌وجود، هر دوی این اعداد به‌صورت باورنکردنی بزرگ هستند. ورودی‌های فعال توسط یک ناحیه تنها یک بخش ناچیزی از تمام ورودی‌های ممکن را نمایش می‌دهد. پس اینکه چگونه یک ناحیه، از ورودی‌های خود یک بازنمایی توزیع‌شده تنک را می‌سازد توصیف خواهیم کرد و خواهیم گفت که تئوری از دست دادن داده‌ها دیگر اثر عملی نخواهد داشت. شکل ۱-۴ چگونگی توزیع و فعال شدن تعداد کمی از سلول‌ها در هر ناحیه برای یک الگو ورودی را نمایش می‌دهد [2].



شکل ۴ - شمایی از یک بازنمایی توزیع‌شده تنک برای یک الگوی ورودی [2]

[۱] sparse distributed representation

چنین ساختار بازنمایی خصوصیات مطلوب متعددی دارد. این بازنمایی در تمام سطوح و ناحیه‌های این حافظه وجود دارد که در بخش‌های بعدی به این موضوع خواهیم پرداخت.

۴.۱.۲. نقش زمان

زمان یک نقش بسیار حیاتی را در یادگیری، استنتاج و پیش‌بینی دارد. بگذارید با استنتاج شروع کنیم. بدون استفاده از زمان، ما نمی‌توانیم چیزی از حس لامسه یا حس شنوایی درک کنیم. برای مثال اگر شما کور باشید و یک شخصی در دستان شما یک سیب قرار دهد، شما قادر خواهید بود که بعد از لحظاتی لمس کردن آن، این شی را شناسایی کنید. در حین اینکه شما دارید اطراف سیب را لمس می‌کنید، اطلاعات لمس شده به‌صورت دائم در حال تغییر هستند. این شی به‌خودی‌خود (یک سیب) یک درک سطح بالا برای شما دارد و همیشه ثابت می‌ماند. درحالی‌که اگر یک سیب در دستان شما قرار داشته باشد و شما اجازه حرکت دستان و انگشتان خود را نداشته باشید، شما به سختی می‌توانید تشخیص دهید که این شی سیب است یا یک میوه دیگر؛ و این برای صوت نیز صحیح است. صداهای ثابت معنای کوچکی را می‌رسانند. یک کلمه شبیه سیب یا صدای گاز زدن یک سیب توسط یک شخص، تنها می‌تواند توسط هزاران صوت سریع دنباله‌داری که در حین زمان شنیده می‌شود به دست آید.

بینایی، برخلاف مورد بالا، یک موضوع مخلوط است. برخلاف لامسه و یا شنوایی، انسان‌ها می‌توانند تصاویر را زمانی که به سرعت در جلوی آن‌ها ظاهر می‌شود با حرکت دادن سریع چشمان خود، بشناسند؛ بنابراین، استنتاج تصویری[۱]، همیشه نیاز به ورودی‌های متغیر با زمان ندارد. درحالی‌که ما برای دیدن، چشمان، سر و اندام خود را حرکت می‌دهیم و همچنین اشیا نیز در حالش چرخیدن در اطراف ما هستند. ولی به‌صورت کلی، در بینایی، شنوایی و لامسه ورودی‌های متغیر با زمان ممکن است وجود داشته باشند و لازم هستند.

همان‌طور که در مورد مسائل عمومی استنتاج و مسائل خاص در مورد استنتاج‌های تصاویر ثابت گفته شد، حالا به مبحث یادگیری می‌پردازیم. در حافظه زمانی سلسله‌مراتبی ورودی‌های متغیر با زمان در حین یادگیری باید داشته باشیم. حتی در بینایی که بحث شناسایی تصاویر ثابت وجود دارد ما باید تغییر تصویر اشیا را در نظر بگیریم تا بگوییم که تصاویر به کدام اشیا نزدیک‌تر هستند. به‌عنوان‌مثال یک سگ را در نظر بگیرید که در حال دویدن در مقابل شماست. تصاویری که از این سگ می‌بینید در هرلحظه به یک شکل و یک الگوی خاص هستند و به‌صورت ریاضی ممکن است این تصاویر دارای تفاوت‌هایی نیز باشند، ولی مغز انسان این الگوهای متفاوت را به این شکل در نظر می‌گیرد که این‌ها دنباله‌ای از مشاهدات متوالی از حرکت سگ است. زمان یک ناظر است که به شما یاد می‌دهد که کدام دنباله‌ها در چه فاصله‌ای باید باهم ترکیب شوند که بتوانید استنتاج انجام دهید.

توجه داشته باشید که این تنها کافی نیست که حسگرهای ورودی متغیر با زمان داشته باشید. توالی الگوهایی که از حسگرهای نامربوط می‌آیند، می‌تواند باعث سردرگمی شود. ورودی‌های متغیر با زمان باید از یک منبع مشترک از دنیا بیایند؛ و همچنین در نظر داشته باشید که هرچند ورودی از حس‌های انسانی به دست می‌آیند ولی به‌طورکلی برای ورودی‌های غیرانسانی مسائل به‌صورت خوبی حل می‌شوند. به‌عنوان‌مثال اگر حافظه زمانی سلسله‌مراتبی قرار باشد که الگوهایی از دمای نیروگاه را شناسایی کند، حسگرهای ورودی باید حرکتی و نویزی هستند که این‌ها حسگرهای انسانی نیستند.

حافظه زمانی سلسله‌مراتبی نیاز دارد تا با داده‌های بسیار زیادی کار کند و آموزش ببیند. به‌عنوان‌مثال زمانی که شما یک سگ را تشخیص می‌دهید، به این دلیل است که شما با دیدن تعداد زیادی از انواع سگ‌ها می‌توانید تعداد زیادی از انواع سگ‌های دیگر را نیز تشخیص دهید و فقط با دیدن یک سگ به این درک بالا از شناخت انواع سگ‌ها

[۱] vision inference

نرسیده‌اید. وظیفه این شبکه این است که از یک سری دنباله‌های زمانی که از دیتاهای ورودی می‌آید آموزش ببیند. این وظیفه‌ی بسیار سختی است به این دلیل که ما دقیقاً نمی‌توانیم بگوییم که اول و آخر این دنباله‌ها کجاست و یا حتی ممکن است این دنباله‌ها همپوشانی داشته باشند و از طرف دیگر ممکن است داده‌های ورودی دارای نویز باشند و درنتیجه کار ما را سخت کنند.

حافظه زمانی سلسله‌مراتبی سعی دارد که بر اساس شناخت این دنباله‌ها که پایه شکل‌گیری پیش‌بینی است، به پیش‌بینی سایر الگوهای مشابه که در آینده وارد سیستم خواهند شد، بپردازد به‌گونه‌ای که الگوی فعلی داده شده به شبکه را با الگوهای از قبل استنتاج شده مقایسه و مشابهات را پیدا کند. ما در ادامه گزارش به توضیح چهار تابع اصلی از این شبکه می‌پردازیم که به این صورت می‌باشند: یادگیری[۱]، استنتاج[۲]، پیش‌بینی[۳] و رفتار[۴]. هر ناحیه حافظه زمانی سلسله‌مراتبی سه تابع اول را انجام می‌دهد. درحالی‌که مورد چهارم –رفتار- مورد متفاوتی است. در مباحث زیست‌شناسی چنین گفته‌شده است که تعدادی زیادی از نواحی قشر مخ، یک نقش و وظیفه را در ساخت رفتار دارند که ما اعتقاد نداریم این موضوع در کاربردهای فعلی ضروری باشد. به همین خاطر در پیاده‌سازی‌هایی تاکنون انجام شده، زیاد سعی نشده است که بحث رفتار در آن گنجانده شود و در این گزارش نیز در این مورد صبحتی نخواهیم کرد [2].

۲.۲. مولفه های اصلی حافظه زمانی سلسله مراتبی

۱.۲.۲. یادگیری

حافظه زمانی سلسله‌مراتبی و نواحی آن با استفاده از الگوها و دنباله‌هایی از الگوهایی که از ورودی‌ها دریافت می‌کند به یادگیری می‌پردازد. نواحی نمی‌دانند که ورودی آمده از حسگرها چیست و چه چیزی را بازنمایی می‌کند و تنها در قلمرو کاملاً آماری کار می‌کنند. ترکیب این ورودی‌ها را به عنوان الگوهای مکانی[۵] (فضایی) می‌شناسیم. این الگوهای مکانی در دنباله‌های زمانی ظاهر می‌شوند که ما آن‌ها را الگوها زمانی[۶] یا دنباله‌های زمانی[۷] می‌نامیم.

اگر ورودی‌های یک ناحیه، حسگرهایی از یک ساختمان را داشته باشد، این ناحیه شاید چنین تشخیص دهد که ترکیب بعضی از ورودی‌هایی از دما و رطوبت را دارد که در شمال ساختمان هستند. این ترکیبات باید متفاوت از ترکیبات سایر حسگرها برای جنوب ساختمان باشند و این ناحیه یاد می‌گیرد که دنباله‌ای از این ترکیبات در هر روزی که می‌گیرد اتفاق می‌افتد.

برای مثال دیگر، اگر ورودی‌های به یک ناحیه، اطلاعاتی مرتبط با خرید از یک فروشگاه را بگیرد، حافظه زمانی سلسله‌مراتبی چنین تشخیص می‌دهد که بعضی از کالاهای خاص در آخر هفته بیشتر فروش می‌روند. سپس این شبکه یاد می‌گیرد که افراد متفاوت در خرید خود، الگوهای مشابهی را دنبال می‌کنند.

همان‌گونه که قبلاً گفته شد، اگر حافظه داده شده به یک ناحیه زیاد باشد، اطلاعات و توانایی‌های آن ناحیه بیشتر خواهد بود و برعکس. به همین دلیل اگر یادگیری یک ناحیه از حافظه زمانی سلسله‌مراتبی ساده باشد،

---

[۱] Learning

[۲] Inference

[۳] Prediction

[۴] Behavior

[۵] spatial patterns

[۶] temporal patterns

[۷] temporal sequences

درنتیجه این شبکه نیاز به سلسله‌مراتب بیشتر دارد تا بتواند استنتاج‌های پیچیده و سطح بالاتری را انجام دهد. این مورد را می‌توان در بینایی انسان دید، جایی که شبکیه چشم انسان اطلاعات را دریافت می‌کند و به نواحی داخل مغز می‌دهد و الگوهای مکانی را برای فضاهای کوچک یاد می‌گیرد و این دنباله‌ها را به‌صورت مداوم به مغز می‌دهد تا نهایتاً مغز این الگوها را ترکیب کند و یک نتیجه‌گیری و استنتاج از این دنباله تصاویر را بدهد. همان‌طور که مشخص است، این روال به‌صورت یک سلسله‌مراتب در حال انجام شدن است [2].

یکی از الگوریتم‌هایی که برای یادگیری در چنین محیط‌های پویایی پیشنهاد شده است الگوریتم نیدلمن-وانچ[1] است؛ اما این الگوریتم نیز دارای ضعف‌هایی است که باعث شده است زیاد به آن توجه نشود. البته در پیاده‌سازی کاربرد شناسایی زبان اشاره که در این گزارش گفته می‌شود، از همین روش به عنوان یک راه کمکی برای بازدهی بیشتر حافظه زمانی سلسله مراتبی استفاده شده است [6].

مشابه یک سیستم زیستی، الگوریتم‌های یادگیری حافظه زمانی سلسله‌مراتبی قادر خواهند بود که عمل یادگیری آنلاین[2] را انجام دهند و آن‌ها به‌صورت مداوم در حال یادگیری از ورودی‌های جدید خود هستند. اینجا نیاز نیست که فاز یادگیری را از فاز استنتاج جدا کنیم، هرچند بعد از یادگیری‌های اضافی، استنتاج بهبود میابد. همان‌طور که ورودی‌ها در حال تغییر هستند، نواحی شبکه نیز به‌تدریج تغییر می‌کنند [2].

بعد از یادگیری اولیه، شبکه می‌تواند به یادگیری ادامه دهد و یا به‌صورت متناوب می‌توانیم یادگیری را غیرفعال کنیم. شبکه قابلیت دیگری که دارد این است که می‌تواند یادگیری را فقط برای سطوح پایین شبکه غیرفعال کند و همچنان برای لایه‌های بالایی شبکه آموزش در حال انجام باشد. معمولاً حافظه زمانی سلسله‌مراتبی در لایه‌های پایین‌تر یادگیری را انجام می‌دهد و بعد از آن با یادگیری جدید سعی در آموزش لایه‌های بالاتر را خواهد داشت. زمانی که شبکه الگوی جدید را ببینید که قبلاً توسط لایه‌های پایین‌تر شبکه دیده نشده بود، سعی در یادگیری این الگوهای جدید می‌کند. ما این روند را در انسان هم مبینیم. یادگیری کلمات جدید در یک زبانی که شما از قبل با آن آشنایی دارید بسیار ساده‌تر است تا زمانی که بخواهید کلمه‌ای را از یک زبان خارجی که با آن آشنایی نداشتید را یاد بگیرید.

حقیقتاً کشف الگوها یک توانایی بسیار باارزش است. ادراک از الگوهای سطح بالایی همچون نوسانات بازار، بیماری‌ها، آب‌وهوا، عملکرد تولید و یا خرابی سیستم‌های پیچیده مثل شبکه‌های قدرت، دارای ارزش بالایی است. بااین‌وجود یادگیری مکانی و زمانی الگوها یک اصل برای استنتاج و پیش‌بینی است [2].

۲.۲.۲. استنتاج

زمانی که حافظه زمانی سلسله‌مراتبی یادگیری خود را انجام داد، می‌تواند از داده‌های ورودی جدید که به آن وارد می‌شود عمل استنتاج را انجام دهد. زمانی که شبکه، ورودی جدید را می‌گیرد این ورودی را با الگوهای مکانی و زمانی قبلی یاد گرفته، تطابق می‌دهد. تصور کنید که شما می‌خواهید یک ملودی را تشخیص بدهید. شما با شنیدن اول نت شاید بتوانید تا حدودی آن را تشخیص دهید. شاید دومین نت هم به شما کمک کند ولی بازهم ممکن است کافی نباشد و به احتمال زیاد باید تعداد نت‌های بیشتری شنیده شود تا ملودی موردنظر پیدا شود. استنتاج در حافظه زمانی سلسله‌مراتبی مشابه این است. شبکه به‌صورت مداوم دنباله ورودی‌ها را می‌گیرد و با الگوها و دنباله‌های از قبل یاد گرفته مقایسه می‌کند. این زمانی است که اگر یک ملودی از اول آغاز شود حافظه می‌تواند تطابق موردنظر را پیدا کند؛ ولی آیا می‌تواند این تشخیص را زمانی انجام دهد که ملودی از هر قسمتی به شبکه داده شود؟ جواب این سوال

[1] Needleman–Wunsch algorithm

[2] on-line learning

را می‌توان در ساختار حافظه شبکه دید که داده‌های به‌صورت توزیع‌شده نگه‌داری می‌شوند و این کار باعث می‌شود که نواحی از این حافظه دنباله‌دار استفاده کنند.

این شبکه به این‌که نحوه ورودی اطلاعات جدید به چه صورت است کاری ندارد و درهرحال الگوی مشابه با ورودی جدید را پیدا می‌کند. برای مثال، زمانی که شما لغت صبحانه را می‌شنوید، این موضوع اهمیت ندارد که این کلمه از زبان یک پیرمرد شنیده شود یا یک جوان، یا اینکه زن است یا مرد، آهسته گفته‌شده باشد یا سریع. بااین‌حال اگر یک شخص این لغت را صدها بار تلفظ کند، این صدا هیچ‌گاه حلزون گوش (تحریک شخص مخاطب) را برای دو بار بر نخواهد انگیخت [2].

این حافظه مانند مغز انسان با چنین مسائلی روبرو است که ورودی‌ها هیچ‌گاه به‌صورت واقعی تکرار نمی‌شوند. درنتیجه این شبکه باید این داده‌های جدید را در حین آموزش و استنتاج اداره کند. یکی از روش‌ها برای حل این مسائل این است که از بازنمایی توزیع‌شده تنک استفاده کنیم. یک خصوصیت کلیدی این روش این است که شما فقط نیاز دارید که یک قسمتی از الگوی ورودی را با یک قسمتی معنی‌دار حافظه تطبیق بدهید [2,3].

۳.۲.۲. پیش‌بینی

هر ناحیه از حافظه زمانی سلسله‌مراتبی دنباله‌ای از الگوها را ذخیره می‌کند. ناحیه به این صورت عمل می‌کند که ورودی جدید را با دنباله‌ی الگوهای ذخیره‌شده مطابقت می‌دهد و از طرفی بر اساس این دنباله‌ها پیش‌بینی می‌کند که ورودی بعدی به چه صورت خواهد بود. ناحیه‌ها در حقیقت این انتقال‌هایی که بین بازنمایی‌های توزیع‌شده تنک انجام می‌شد را ذخیره می‌کنند. برای بعضی از مثال‌ها این انتقال‌ها شبیه یک دنباله خطی می‌باشند، مانند نت‌های یک ملودی، ولی در مسائل کلی، تعداد زیادی از ورودی‌های که در آینده می‌آید را می‌تواند به‌صورت هم‌زمان پیش‌بینی کند. یک ناحیه در این شبکه، پیش‌بینی‌های متفاوتی بر اساس مفهوم (مفهومی که الگوهای قبلی و فعلی، در محیط کنونی دارند) خواهد داشت که می‌تواند که در زمان عقب برود (رجوع به دنباله‌ها و الگوهای قبلی). مقدار زیادی از حافظه در این شبکه به حافظه متوالی و یا نگه‌داری انتقال بین الگوهای مکانی اختصاص داده شده است. در زیر تعدادی از خصوصیات کلیدی تخمین در این شبکه را گفته‌ایم.

- **تداوم پیش‌بینی**

بدون اینکه شما نسبت به این موضوع آگاه باشید، شما در حال پیش‌بینی هستید. حافظه زمانی سلسله‌مراتبی هم شبیه انسان در حال پیش‌بینی است. مثلاً زمانی که شما به یک موسیقی گوش می‌دهید ذهن شما به‌صورت متوالی در حال پیش‌بینی نت‌ها و کلمات گفته‌شده در آینده است. یا زمانی که شما از پله پایین می‌آیید شما در حال پیش‌بینی این هستید که پای شما چه زمانی به پله بعدی مارسد؛ و یا زمانی که در حال دیدن ورزش بیس‌بال از تلویزیون هستید ذهن شما پیش‌بینی می‌کند که توپ چه زمانی به منحنی ضربه توپ خواهد رسید. در این شبکه پیش‌بینی و استنتاج به‌صورت کلی یک عملکرد محسوب می‌شود. پیش‌بینی یک مرحله جدا نیست، ولی به‌صورت ضمنی در وظایف ناحیه‌ها قرار داده شده است.

- **پیش‌بینی در هر ناحیه در هر سطح از سلسله‌مراتب به وجود می‌آید**

در این شبکه، در هر سطح و در هر ناحیه پیش‌بینی به‌صورت مداوم در حال انجام شدن است. به‌عنوان‌مثال سطوح پایین شبکه در حال پیش‌بینی آواهای بعدی گفتار هستند ولی در سطوح بالاتر، پیش‌بینی کلمات بعدی انجام می‌شود.

- **پیش‌بینی حساس به مفهوم و محیط است**

پیش‌بینی بر این اساس است که در گذشته چه پیش آمده است و هم‌اکنون چه چیزی در حال اتفاق افتادن است؛ بنابراین یک ورودی، پیش‌بینی متفاوتی را بر اساس مفهوم قبلی تخمین میزند. حافظه زمانی سلسله‌مراتبی یاد می‌گیرد که تا چه حد از مفاهیم قبلی که نیاز است، استفاده و مفاهیم را به‌صورت کوتاه‌مدت و یا بلندمدت نگه‌داری کند. این توانایی را به‌عنوان حافظه مرتبه متغیر[1] می‌نامیم. به‌عنوان‌مثال فکر کردن در مورد یک صحبت حفظ‌شده، مانند آدرس یک شرکت. برای پیش‌بینی کلمه بعدی فقط کافی است که کلمه فعلی را بدانیم؛ و با آمدن هر کلمه شما می‌توانید کلمه بعدی را پیش‌بینی کنید.

- **تخمین باعث پایداری می‌شود**

خروجی یک ناحیه، پیش‌بینی انجام شده توسط آن است. یکی از خصوصیات حافظه زمانی سلسله‌مراتبی این است که خروجی ناحیه‌ها بسیار پایدارتر می‌شوند زمانی که به سمت سطوح بالاتر شبکه پیش می‌رویم. این خصوصیت از اینکه یک ناحیه به چه صورت پیشی بینی انجام می‌دهد نشأت می‌گیرد. حافظه زمانی سلسله‌مراتبی فقط پیش‌بینی زمان آینده نزدیک را انجام نمی‌دهد بلکه این شبکه می‌تواند پیش‌بینی‌های چند زمان بعدی که در پیش رو هست را نیز داشته باشد. فرض کنید یک ناحیه می‌تواند ۵ مرحله پیش‌بینی انجام دهد. زمانی که یک ورودی جدید مارسد، پیش‌بینی مرحله اول این ناحیه تغییر خواهد کرد ولی ۴ مرحله دیگر بدون تغییر خواهند ماند. درنتیجه حتی اگر یک ورودی جدید که کاملاً متفاوت از قبل بوده باشد را به این ناحیه اعمال کنیم فقط یک قسمت از خروجی تغییر خواهد کرد و این باعث می‌شود که شبکه به‌صورت پایدار پیش برود. این خصوصیت باعث بازتاب تجربیات و آزمایش‌های ما در دنیای واقعی را دارد. جایی که مفاهیم سطح بالا مانند اسم یک آهنگ کمتر از مفاهیم سطح پایین مثل نت‌های آن آهنگ تغییر می‌کنند.

- **پیش‌بینی به ما می‌گوید که آیا ورودی جدید قابل‌انتظار بوده است یا نه**

هر ناحیه از حافظه زمانی سلسله‌مراتبی یک تشخیص‌دهنده ورودی‌های جدید است. به خاطر اینکه هر ناحیه پیش‌بینی می‌کند که در آینده چه اتفاقی می‌افتد، از طرفی این شبکه می‌تواند متوجه شود که چه زمانی اتفاقات غیرمنتظره‌ای رخ داده است. این حافظه می‌تواند خیلی از ورودی‌های بعدی را به‌صورت همزمان پیش‌بینی کند و نه فقط یک ورودی را. پس این شبکه ممکن است نتواند به خوبی پیش‌بینی‌هایی را انجام دهد، اما اگر ورودی آمده با پیش‌بینی‌های انجام شده توسط نواحی مطابقت نداشت، شبکه چنین برداشت می‌کند که یک سری رفتارهای ناهنجار در حال اتفاق افتادن است.

- **پیش‌بینی می‌تواند به شناخت و غلبه بیشتر در مقابل نویز کمک کند**

زمانی که حافظه زمانی سلسله‌مراتبی ورودی‌های آینده را پیش‌بینی می‌کند، این پیشی بینی می‌تواند سیستم را به سمت استنتاج اینکه چه چیزی تخمین زده شده است سوق دهد. به‌عنوان‌مثال اگر یک حافظه زمانی سلسله‌مراتبی در حال پردازش کردن زبان گفتاری باشد، این شبکه پیش‌بینی خواهد کرد که چه صداها، کلمات و ایده‌هایی توسط شخص اداکننده در آینده گفته می‌شود. این پیش‌بینی باعث می‌شود که سیستم، بازیابی اطلاعات ازدست‌رفته را نیز انجام دهد. زمانی که یک صدای مبهم به سیستم برسد، شبکه این صدای مبهم را با آنچه پیش‌بینی کرده بود تفسیر و به شکل صحیح آن را بازیابی می‌کند و از طرفی این پیش‌بینی به استنتاج در حضور نویز نیز کمک خواهد کرد. در یک ناحیه مواردی همچون حافظه متوالی، استنتاج و پیش‌بینی تقریباً به‌صورت مجتمع هستند. این وظایف، وظایف اصلی از یک ناحیه در این شبکه می‌باشند [2].

[1] variable order memory

۴.۲.۲. رفتار

رفتار ما تحت تأثیر چیزهایی که ما مشاهده و یا درک می‌کنیم قرار می‌گیرد. به محض اینکه ما چشمان خود را حرکت می‌دهیم، شبکیه چشم ما، ورودی‌های متغیر جدیدی را دریافت می‌کند. حرکت اعضای بدن و یا انگشتان باعث تغییر در حس لامسه ما می‌شود که به مغز مارسد؛ و درنتیجه تمام فعالیت‌های ما بر اساس چیزی که ما حس می‌کنیم تغییر می‌کند. ورودی‌های حسگرها و رفتار حرکتی ما با هم عجین شده‌اند.

برای دهه‌ها تفکر غالب به این صورت بوده است که تنها یک ناحیه خاص در مغز انسان است که دستورات از آنجا ارسال می‌شود. ولی به‌مرورزمان مشخص شد که هر قسمت از مغز دارای محرک‌های دستوری است؛ حتی نواحی حسگر در سطوح پایین. این بدان معناست که تمام نواحی در قشر مخ دارای حسگرها و توابع محرک هستند.

محققان انتظار دارند که این محرک‌ها را در هر قسمت از پیاده‌سازی‌های حافظه زمانی سلسله‌مراتبی قرار بدهند. تمام پیاده‌سازی‌های حافظه زمانی سلسله‌مراتبی تاکنون فقط دارای حسگرها بوده‌اند و مؤلفه‌های محرک در آن هنوز پیاده نشده است [2].

۳.۲. پیشرفت‌های به‌سوی پیاده‌سازی حافظه زمانی سلسله‌مراتبی

تاکنون پیشرفت‌ها و تلاش‌های زیادی برای پیاده‌سازی تئوری‌های حافظه زمانی سلسله‌مراتبی در کاربردهای تکنولوژی شده است؛ و همچنین نسخه‌های متفاوتی از حافظه زمانی سلسله‌مراتبی پیاده‌سازی و تست گردیده و یک پیاده‌سازی پایه برای این شبکه پیدا شده است. تاکنون تعداد زیادی از تئوری‌ها و اصولی که در حافظه زمانی سلسله‌مراتبی به‌صورت تئوری بحث شده است، هنوز پیاده‌سازی نشده‌اند مانند توجه، بازخورد بین نواحی، زمان‌بندی خاص و مبحث رفتار حسی که محققان در حال کار بر روی این موارد می‌باشند [2,1].

در مورد پیاده‌سازی این حافظه تا کنون کارهایی انجام شده است ولی به دلیل تازگی مباحث و پیچیدگی پیاده‌سازی آن (به دلیل پیچیدگی‌های ساختاری)، مقالات و یا کاربردهای آن هنوز چشم‌گیر نبوده است؛ اما پیاده‌سازی‌هایی همچون شناسایی زبان اشاره، بازشناسی صوت[۱]، شناسایی الگو در شبکه‌های ادهاک [11] و یا شناسایی الگو در تصاویر ویدئویی [9] اشاره کرد.

۳. الگوریتم یادگیری قشر مغزی حافظه زمانی سلسله‌مراتبی

در این فصل به محبت الگوریتم یادگیری حافظه زمانی سلسله‌مراتبی در ناحیه‌ها می‌پردازیم و در فصل‌های ۴ و ۵ پیاده‌سازی این الگوریتم را با شبه دستورها نمایش می‌دهیم. در این فصل بیشتر با مفاهیم الگوریتم برخورد می‌کنیم.

۱.۳. اصطلاحات

قبل از اینکه این بخش را شروع کنیم، گفتن یک سری از اصطلاحاتی که در این شبکه وجود دارد می‌تواند مفید باشد. برای توصیف شبکه‌های حافظه زمانی سلسله‌مراتبی از زبان علوم عصب‌شناسی[۲] استفاده می‌شود. برای کسی که با چنین اصطلاحاتی آشنا نیست، مشکلی نخواهد بود ولی برای کسانی که با چنین اصطلاحاتی آشنا هستند ممکن مشکلاتی پیش بیاید به دلیل استفاده بعضی از این کلمات در بیان بعضی از قسمت‌های شبکه باعث سردرگمی مخاطب می‌شود؛ زیرا ممکن است در پیاده‌سازی الگوریتم و تعاریفی که شده است تفاوت و انحرافاتی از علم

[۱] Speech Recognition

[۲] neuroscience

زیست‌شناسی وجود داشته باشد به همین دلیل در زیر این اصطلاحات را تعریف کرده و این تفاوت‌ها را نیز اشاره خواهیم کرد [2].

- **حالت‌های سلول**

سلول‌های حافظه زمانی سلسله‌مراتبی هرکدام دارای سه حالت خروجی هستند، فعال از ورودی پیش‌خور، فعال از ورودی عرضی (که به‌عنوان پیش‌بینی مطرح شد) و غیرفعال. اولین حالت خروجی مربوط است به فعالیت‌های کوتاه پشت سرهم که به‌صورت بالقوه در نورون وجود دارد. دومین خروجی مربوط به یک نرخ آهسته و پایدار از فعالیت‌های نورون است. در این شبکه لازم دیده نشده است که این دو خروجی را به عنوان یک خروجی واحد در نظر بگیریم. استفاده از بازنمایی توزیع‌شده باعث می‌شود که نیاز به مدل‌های با نرخ‌های فعالیت عددی نداشته باشیم.

- **قسمت‌های دندریتی**

سلول‌های حافظه زمانی سلسله‌مراتبی یک مدل دندریتی[۱] نسبتاً واقع‌بینانه هستند (و پیچیده). به‌صورت تئوری هر سلول یک بخش نزدیک به مبدأ دارد و تعداد زیادی و یا دو عدد دندریت دور از مبدأ. بخش نزدیک به مبدأ (دندریت پروگزیمال[۲]) ورودی‌های پیش‌خور را دریافت می‌کند و دندریت‌های دیستال[۳] که از مبدأ دور هستند ورودی‌ها را به‌صورت عرضی از سلول‌های هم‌جوار می‌گیرند. یک کلاس از سلول‌های بازدارنده، تمام سلول‌ها در یک ستون را مجبور می‌کند که به یک ورودی پیش‌خور مشابه پاسخ بدهند. برای راحتی، دندریت پروگزیمال از سلول‌های حافظه زمانی سلسله‌مراتبی برداشته شده است و به جای آن یک بخش دندریت اشتراکی به ازای هر ستون از سلول‌ها قرار می‌دهیم. تابع متمرکز کننده مکانی[۴] (که در بخش‌های بعدی توضیح داده می‌شود) در سطح این ستون‌ها، روی این دندریت‌های اشتراکی عمل می‌کند. توابع متمرکز کننده زمانی[۵]، روی دندریت‌های دیستال در سطح سلول‌های انفرادی در ستون‌ها فعالت می‌کنند. این ساده‌سازی یک قابلیت مشابه را برای این اتصالات در هر نقطه از سلول نتیجه می‌دهد، ولی در زیست‌شناسی هیچ تساوی بین دندریت‌های متصل به یک ستون وجود ندارد.

- **سیناپس**

سیناپس‌های[۶] حافظه زمانی سلسله‌مراتبی دارای وزن‌های باینری هستند. سیناپس‌های زیستی دارای وزن‌هایی نیمه تصادفی متفاوتی هستند. نشان داده شده است که یک نورون زیستی[۷] (نورون واقعی در مغز) نمی‌تواند به وزن‌های سیناپسی دقیق اعتماد کند. استفاده از بازنمایی توزیع‌شده در حافظه زمانی سلسله‌مراتبی به مدل فعالیت‌های دندریتی ما این امکان را اضافه می‌کند که بتوانیم وزن‌های باینری را به این سیناپس‌ها بدهیم بدون اینکه هیچ اثر بدی در بازنمایی و یا عملکرد شبکه به وجود بیاید. برای مدل کردن تشکیل و از بین بردن این سیناپس‌ها از دو مفهوم اضافی مربوط به علوم عصب‌شناسی استفاده شده است. یکی از آن‌ها سیناپس بالقوه (سیناپس‌هایی دارای بار الکتریکی) است. این موضوع چنین می‌گوید که اگر تمام آکسون‌ها[۸] به حد کافی نزدیک به یک قسمت دندریت

---

[۱] dendrite model

[۲] proximal dendrite

[۳] distal dendrite

[۴] spatial pooler function

[۵] temporal pooler function

[۶] synapses

[۷] biological neuron

[۸] axon

باشند، می‌توانند به‌صورت بالقوه یک سیناپس را تشکیل دهند. دومین مفهوم پایداری[1] است. این یک عدد است که به هر سیناپس فعال می‌دهیم. این عدد میزان قوی بودن اتصال بین آکسون و دندریت را می‌رساند. به‌صورت بیولوژیکی، این مقدار می‌تواند از نداشتن اتصال تا شروع تشکیل یک سیناپس جدید باشد -اما هنوز اتصال برقرار نشده است- و از یک سیناپس با حداقل پایداری اتصال تا یک سیناپس کاملاً متصل. این مقدار بین صفر تا یک است. یادگیری شامل کم کردن و زیاد کردن این قسمت نیز خواهد بود. همان‌گونه که گفته شد مقدار پایداری سیناپس در یک محدوده است به این صورت که اگر مقدار پایداری یک سیناپس از یک حد بیشتر شد، مقدار یک و اگر از یک حدی کمتر شد، مقدار آن صفر می‌شود [2].

۲.۳. مرور کلی بر عملکرد شبکه

فرض کنید که شما یک ناحیه از حافظه زمانی سلسله‌مراتبی هستید. ورودی‌های شما شامل هزار و ده‌ها هزار از بیت‌ها است. این اطلاعات می‌تواند از حسگرها و یا از سطوح دیگر شبکه آمده باشد. آن‌ها می‌توانند به صورت‌های پیچیده‌ای خاموش و یا روشن شوند. شما چه فرضیاتی را در مورد این ورودی‌ها می‌گیرید؟

ما در یک شکل ساده در این مورد صحبت خواهیم کرد. هر ناحیه حافظه زمانی سلسله‌مراتبی برای تعدادی الگوی معمول در این ورودی‌ها جستجو و سپس از دنباله الگوهای ورودی شروع به یادگیری می‌کند. سپس بر اساس حافظه‌هایی که وجود دارد هر ناحیه یک پیش‌بینی انجام می‌دهد و به لایه‌های بالاتر و کناری می‌دهد. این تعریف سطح بالا به نظر ساده می‌آید اما در واقعیت اتفاقات خیلی زیاد می‌افتد که در سه قسمت زیر به آن می‌پردازیم [2].

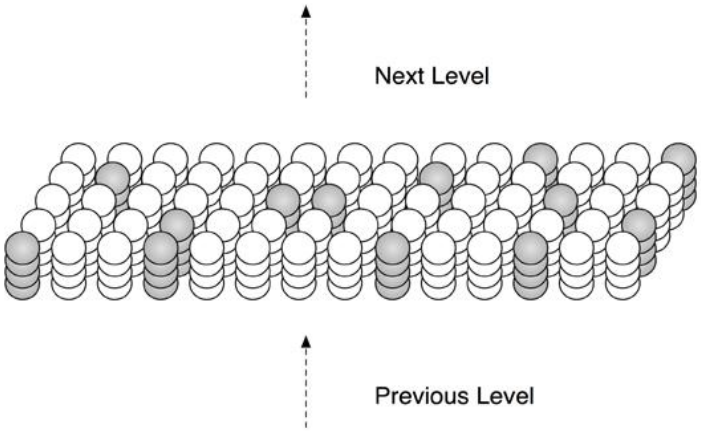
۱.۲.۳. تشکیل یک بازنمایی توزیع‌شده تنک از ورودی

زمانی که ورودی‌های به یک ناحیه را در نظر می‌گیریم ممکن است این تعداد ورودی بسیار زیاد باشد که در مغز همان آکسون‌های یک نورون هستند. در هرلحظه از زمان این ورودی‌ها می‌توانند فعال یا غیرفعال شوند. به‌طورمعمول ورودی‌های فعال می‌تواند بین ۰ تا ۶۰ درصد باشند. اولین کاری یک ناحیه حافظه زمانی سلسله‌مراتبی باید انجام دهد این است که یک بازنمایی تنک برای آن بازسازی کند که به‌عنوان‌مثال اگر ۴۰ درصد از این ورودی‌ها فعال باشند، تنها ۲ درصد از آن‌ها در بازنمایی جدید همچنان فعال بمانند.

یک ناحیه به‌صورت منطقی شامل یک دسته از ستون‌هاست و هر ستون شامل یک یا چند سلول است. ستون‌ها به‌صورت منطقی به‌صورت دوبعدی مرتب شده‌اند اما همیشه الزاماً این‌طور نیست. هر ستون در یک ناحیه به یک مجموعه خاص از ورودها وصل شده است. (معمولاً این اتصالات همپوشانی خواهند داشت ولی نه دقیقاً مشابه زیرمجموعه‌ای از بیت‌های ورودی). پس درنتیجه الگوهای ورودی متفاوت، در سطوح متفاوت از ستون‌ها فعالیت خواهند داشت. ستون‌هایی که بیشترین فعالیت را دارند، سایر ستون‌هایی که کمتر فعال هستند را مانع می‌شوند (غیرفعال می‌کنند). این جلوگیری می‌تواند در یک ناحیه در اطراف آن ستون فعال، باشد. بازنمایی تنک از ورودی‌ها، به واسطه ستون‌هایی که فعال هستند و یا بعد از انجام بازدارندگی (توسط ستون‌های فعال) غیرفعال شده‌اند، رمزنگاری می‌شوند. تابع بازدارندگی را به‌گونه‌ای تعریف می‌کنیم که تعداد ستون‌های فعال به یک درصد نسبی ثابتی تبدیل شوند حتی اگر تعداد واحدهای فعال ورودی مقدار فراوانی باشند. در شکل ۳-۵ قسمتی از یک ناحیه از شبکه را نشان داده‌ایم. در این شکل، ستون‌هایی که به رنگ خاکستری نشان داده‌شده‌اند، ستون‌هایی هستند که به واسطه ورودی‌ها و بعد از انجام عملیات بازدارندگی فعال مانده‌اند. ستون‌هایی که مقدار فعالیت بیشتری دارند با بازدارندگی

[1] permanence

بیشتر ستون‌های با فعالیت کمتر، سعی در فعال ماندن خواهند داشت و این امر باعث می‌شود که فقط درصد کمی از ستون‌ها در یک ناحیه فعال باشند.



شکل ۵ - نمایش بازنمایی تنک ستون‌های فعال [2]

فرض کنید که ورودی تغییر می‌کند. این تغییر به این صورت است که ورودی‌های فعال و غیرفعال تغییر می‌کنند. اگر تعداد ورودی‌هایی که تغییر می‌کنند کم باشد، ستون‌های فعال تغییری نخواهند کرد زیرا تغییرات ورودی‌ها معمولاً در هر ستون به مقدار تعداد کمی، کم یا زیاد می‌شوند و تأثیری در ستون‌های فعال تنک شده فعلی نخواهند داشت؛ بنابراین می‌توان الگوهای ورودی مشابه را (حتی اگر مقادیر نسبت قابل‌توجهی ورودی فعال داشته باشند) به ستون‌های فعال جاری نگاشت کنیم. پایداری به این بستگی دارد که تا چه حد ورودی‌های فعال (اتصالات فعال) به آن وجود دارد. این اتصالاتی که توسط حافظه زمانی سلسله‌مراتبی یاد گرفته می‌شوند را به‌صورت مفصل در بخش‌های بعدی توضیح خواهیم داد.

این مرحله (شامل یادگیری اتصالات به هر ستون از یک زیرمجموعه از ورودی‌ها، مشخص کردن سطح ورودی‌ها برای هر ستون و استفاده از توابع بازدارندگی برای تنک‌سازی) را به متمرکز کننده مکانی می‌شناسیم. این اصطلاح به این معناست که الگوهایی که به‌صورت مکانی مشابه هستند (به این معنی که آن‌ها تعداد زیادی از بیت‌های فعال را اشتراک می‌گذارند) یکی می‌شوند (به این معنی که آن‌ها با هم در یک بازنمایی در یک گروه قرار می‌گیرند) [2].

۲.۲.۳. تشکیل یک بازنمایی از ورودی بر اساس مفهوم ورودی قبلی

تابع بعدی به این صورت است که نمایش ستونی در هر ناحیه را تبدیل به یک بازنمایی جدید می‌کند که شامل حالت یا مفهومی از گذشته باشد. این بازنمایی جدید به این صورت است که یک زیرمجموعه‌ای از سلول‌ها را در هر ستون فعال کنیم به‌صورت معمول فقط یک سلول در هر ستون. (شکل ۳-۶)

فرض کنید دو جمله گفته‌شده را می‌شنویم. جمله اول I ate pear (من یک گلابی را خوردم) و جمله دوم I have eight pears (من هشت گلابی دارم). لغات خوردن[۱] و هشت[۲] در انگلیسی از لحاظ آوایی مشابه هستند. ما مطمئن هستیم که نورون‌های مغز به این قسمت از جمله‌ها که مشابه هستند عکس‌العمل نشان می‌دهد و از طرفی یقین داریم که در نقطه بعدی، نورون‌هایی که به چنین مواردی عکس‌العمل نشان می‌دهند نسبت به حالت قبلی (نورون‌هایی که قبلاً به چنین موردی واکنش نشان داده بودند) متفاوت خواهند بود. بازنمایی کلمات ate و eight

[۱] ate

[۲] eight

برای زمان یکی از جملات I ate و I have eight را می‌شنویم متفاوت خواهد بود. تصور کنید که شما دو جمله I ate pear و I have eight pears را حفظ کرده‌اید. با شنیدن I ate پیش‌بینی ذهن شما متفاوت از حالتی خواهد بود که شما I have eight را می‌شنوید. باید بعد از شنیدن این دو جمله بازنمایی متفاوت داخلی وجود داشته باشد.

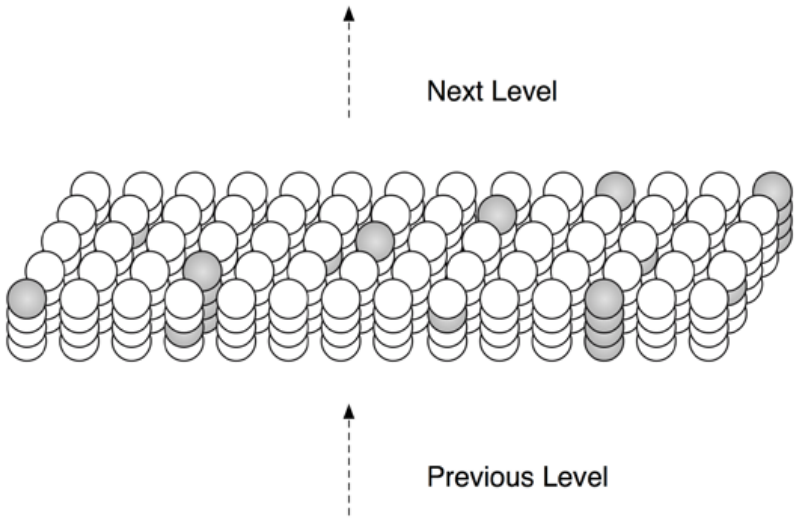
این قاعده رمزنگاری متفاوت، برای یک ورودی در مفهوم‌های (زمینه‌های) مختلف، یک ویژگی کلی از ادراک و واکنش و یکی از مهم‌ترین توابعی است که در یک ناحیه از حافظه زمانی سلسله‌مراتبی پیاده‌سازی شده است. به‌جرئت می‌توان تأکید کرد که این قابلت اهمیت فوق‌العاده‌ای دارد.

هر ستون در هر ناحیه از حافظه زمانی سلسله‌مراتبی شامل تعدادی سلول است و تمام این سلول‌ها یک ورودی پیش‌خور مشابه را دریافت می‌کنند که هر سلول می‌تواند فعال یا غیرفعال باشد. با انتخاب متفاوت از سلول‌های فعال در هر ستون فعال، ما می‌توانیم ورودی‌های کاملاً مشابه را به‌صورت متفاوت برای مفاهیم متفاوت پیاده کنیم. یک مثال خاص می‌تواند به فهم این موضوع کمک کند.

تصور کنید که هر ستون دارای ۴ سلول است و بازنمایی از هر ورودی شامل ۱۰۰ ستون فعال است. اگر در هرلحظه فقط یک سلول از هر ستون بتواند فعال باشد، ما می‌توانیم ۴ به توان ۱۰۰ بازنمایی از این ورودی را داشته باشیم. برای یک ورودی دقیقاً مشابه، همیشه این ۱۰۰ ستون فعال نتیجه می‌شود ولی برای مفاهیم دیگر، ۱۰۰ ستون دیگر فعال خواهند شد. به این شکل ما می‌توانیم برای ورودی‌های مشابه، بازنمایی‌های متفاوتی داشته باشیم؛ اما این بازنمایی‌ها چقدر می‌توانند خاص باشند؟ به‌طور تقریبی برای تمامی جفت انتخاب‌ها از آن ۴ به توان ۱۰۰ الگوی ممکن، حدود ۲۵ سلول دارای همپوشانی خواهند بود. پس برای یک ورودی با دو مفهوم متفاوت ما حدود ۲۵ سلول مشترک داریم و ۷۵ سلول متفاوت که آن‌ها را از هم قابل تمایز می‌کند.

روند عمومی یک ناحیه به این صورت است که زمانی که یک ستون فعال می‌شود، به تمام سلول‌های آن ستون نگاه می‌کند. اگر یک یا چند سلول در آن ستون در حالت پیش‌بینی قرار گرفته باشند، فقط آن سلول‌ها فعال می‌شوند. اگر هیچ سلولی در حالت پیش‌بینی نباشد تمام سلول‌های آن ستون فعال خواهند شد. حتی می‌توان به این مورد نیز فکر کرد که اگر یک الگوی ورودی مورد انتظار باشد سپس سیستم آن انتظار را به وسیله فعال کردن سلول‌هایی که در حالت پیش‌بینی باشند، تأیید می‌کند. اگر الگوی ورودی غیرمنتظره وارد سیستم شود، آنگاه سیستم تمام سلول‌های آن ستون را فعال می‌کند. در مورد جمله آخر چنین می‌توان گفت که ورودی جدید غیرمنتظره بوده است پس تمام بازنمایی‌های ممکن می‌توانند برای این ورودی صحیح باشند.

اگر حالت قبلی وجود نداشت و درنتیجه هیچ پیش‌بینی و یا مفهومی وجود نداشت، تمام سلول‌های داخل ستون فعال خواهند شد، البته زمانی که ستون مربوطه فعال شود. این سناریو شبیه مواقعی است که شما برای اولین بار به نت‌های یک آهنگ گوش می‌دهید. بدون مفهوم، شما معمولاً قادر نخواهید بود که اتفاقاتی که قرار است در آینده اتفاق بیفتد را پیش‌بینی کنید. به همین دلیل تمام موارد ممکن قابل دسترس هستند. اگر حالات قبلی وجود داشته باشند ولی ورودی آمده با هیچ‌کدام از مواردی که انتظار داشتیم مطابق نباشد، تمام سلول‌های داخل ستون فعال به حالت فعال درمی‌آیند. این تصمیم در مورد یک ستون که بر اساس ستون پایه است انجام می‌شود بنابراین یک پیش‌بینی موفق یا ناموفق هرگز یک اتفاق یا حالت همه یا هیچ نخواهد بود. شکل ۳-۶ بازنمایی انجام شده برای یک مفهوم را نشان داده است.



شکل ۶ – بازنمایی مفهوم الگوهای متوالی با استفاده از فعال و غیرفعال کردن سلول‌های موجود در ستون‌ها [2]

همان‌گونه که در قسمت واژه‌ها در بالا گفته شد، سلول‌ها می‌توانند در سه حالت باشند. اگر یک سلول توسط یک ورودی پیش‌خور فعال شده باشد ما فقط به‌اصطلاح فعال میگوییم. اگر سلول توسط اتصالات عرضی فعال شده باشد میگوییم که سلول در حالت پیش‌بینی قرار گرفته است[2].

۳.۲.۳. تشکیل یک پیش‌بینی بر اساس ورودی فعلی و ورودی‌های قبلی

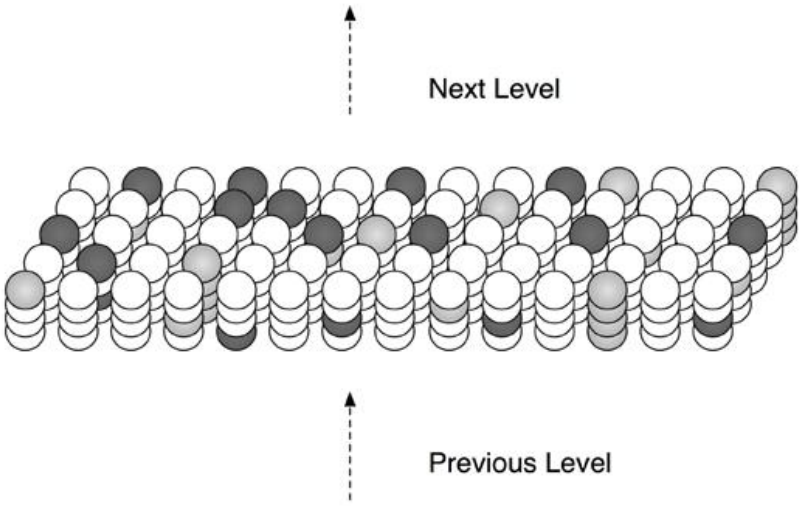
مرحله پایانی برای فعالیت‌های یک ناحیه مبحث پیش‌بینی است که این پیش‌بینی را بر اساس بازنمایی که در مرحله قبل (۲.۲.۳) گفته شد انجام می‌دهد که در آینده چه اتفاقی قرار است بیفتد که آن را بر اساس مفاهیمی که از ورودی‌های قبلی به دست آورده است بیان می‌کند.

زمانی که یک ناحیه یک پیش‌بینی را تولید می‌کند، تمام سلول‌هایی که می‌توانند در طول ورودی‌های آینده پیش‌خور فعال باشند را، فعال می‌کند. به خاطر اینکه بازنمایی در یک ناحیه به‌صورت تنک است، پیش‌بینی‌های چندتایی می‌تواند در یک زمان انجام شود. به‌عنوان‌مثال اگر فقط ۲ درصد از ستون‌ها برای یک ورودی فعال باشند، شما می‌تواند انتظار داشته باشید که ۱۰ پیش‌بینی متفاوت حاصل از ۲۰ درصد از ستون‌هایی که یک سلول پیش‌بینی دارند تولید شود. یا اینکه ۲۰ پیش‌بینی متفاوت حاصل از ۴۰ درصد از ستون‌هایی که یک سلول فعال دارند تولید شود. اگر هر ستون دارای ۴ سلول باشد با یک سلول فعال در هرلحظه، درنتیجه ۱۰ درصد از سلول‌ها در حالت پیش‌بینی خواهند بود.

در بخش بعدی گفته می‌شود که در یک بازنمایی توزیع‌شده تنک بااینکه پیش‌بینی‌های متفاوت با یکدیگر ادغام می‌شوند، یک ناحیه می‌تواند با قطعیت بالا بفهمد که آیا یک ورودی خاص پیش‌بینی شده است یا خیر.

یک ناحیه به چه صورت پیش‌بینی انجام می‌دهد؟ در حین اینکه ورودی‌ها تغییر می‌کنند، سلول‌ها و ستون‌ها نیز بر اساس ورودی‌ها فعال و یا غیرفعال می‌شوند. زمانی که یک سلول فعال می‌شود، این سلول ارتباطی با یک مجموعه‌ای از سلول‌های نزدیک می‌سازد که در لحظات قبلی فعال شده بودند. این ارتباط می‌تواند به‌صورت سریع و یا به‌صورت آهسته شکل بگیرد که بر اساس نرخ یادگیری که به سیستم وارد شده است می‌تواند متفاوت باشد. بعد از این تمام سلول‌ها نیاز دارند تا برای فعالیت‌های همزمان به این اتصالات نگاه کنند. اگر یک اتصال فعال شده باشد، سلول چنین در نظر می‌گیرد که این ارتباط به‌صورت کوتاهی فعال شده است به همین خاطر به حالت پیش‌بینی می‌رود؛ بنابراین فعال‌سازی پیش‌خور یک مجموعه از سلول‌ها، به فعال‌سازی بقیه مجموعه سلول‌ها که به‌صورت

معمول دنبال می‌شوند، منجر می‌شود. به‌عنوان‌مثال: زمانی که شما یک آهنگ را می‌شناسید و در هنگام شنیدن آن شروع به پیش‌بینی نت‌های بعدی آن می‌کنید. شکل ۳-۷ نشان می‌دهد که در هرلحظه ما می‌توانیم هم سلول‌های فعال (فعال شده توسط ورودی‌های پیش‌خور از لایه‌های دیگر) و سلول‌های پیش‌بینی (فعال شده توسط ورودی‌های عرضی در همان لایه) است.



شکل ۷ – نمایش سلول‌های فعال (خاکستری روشن) و سلول‌های پیش‌بینی (خاکستری تیره) [2]

به‌طور خلاصه زمانی که یک ورودی جدید می‌آید، شبکه به سمت فعال کردن یک مجموعه تنک از ستون‌ها می‌رود. زمانی که یک یا چند سلول در هر ستون فعال می‌شود، باعث می‌شود که بقیه سلول‌ها به یک حالت پیش‌بینی وارد شوند که این تغییر حالت بر اساس اتصالات یاد گرفته‌شده بین سلول‌های داخل آن ناحیه است. سلول‌های فعال شده با ارتباطات در داخل یک ناحیه، یک پیش‌بینی از اینکه در آینده چه چیزی قرار است اتفاق بیفتد را درست می‌کنند. زمانی که ورودی پیش‌خور بعدی از راه برسد، این ورودی یک مجموعه تنک دیگر از ستون‌ها را فعال می‌کند. اگر یک ستون که به تازگی فعال شده است به‌صورت غیرمنتظره باشد، به این معنی است که این سلول توسط هیچ سلولی پیش‌بینی نشده بوده است و این باعث می‌شود که تمام سلول‌های آن ستون فعال شوند. اگر ستون به تازگی فعال شده، دارای یک یا چند سلول پیش‌بینی شده باشد، تنها آن سلول‌ها فعال می‌شوند. خروجی این ناحیه فعالیت تمام سلول‌های این ناحیه است، شامل سلول‌های که به خاطر ورودی پیش‌خور فعال شده‌اند و یا سلول‌های که در حالت پیش‌گویی فعال شده‌اند.

در کل چنین می‌توان گفت که این شبکه فقط به پیش‌بینی یک زمان در آینده بسنده نمی‌کند و دنباله‌ای از زمان‌های آینده را بر اساس ورودی‌های فعلی پیش‌بینی می‌کند. پیش‌بینی همان مجموع فعالیت‌های یک ناحیه اعم از فعال بودن ستون‌ها و سلول‌ها و نیز حالات پیش‌بینی سلول‌هاست. از طرفی تأکید کردیم که پایداری در سطوح لایه بالای این شبکه یکی از ویژگی‌هایی است که باعث می‌شود خروجی‌های شبکه زیاد تغییر نکنند و پایداری داشته باشند، با توجه به اینکه ممکن است ورودی به‌طور کامل عوض شود.

به‌عنوان‌مثال فرض کنید که این شبکه در حال شنیدن یک موسیقی است و می‌تواند تا چهار مرحله زمانی بعدی را پیش‌بینی کند. تصور کنید که موسیقی شامل نت‌های A,B,C,D,E,F,G هستند. زمانی که شما نت‌های اول و دوم را می‌شنوید، شبکه پیش‌بینی می‌کند که نت‌های بعدی به‌صورت B,C,D,E,F هستند که B ورودی فعلی است که توسط تعدادی از سلول‌ها در حال نمایش هستند و C,D,E,F حالات پیش‌بینی شده توسط این شبکه. اگر ورودی بعدی که C هست بیاید، شبکه به چنین صورتی پیش‌بینی را انجام می‌دهد: C,D,E,F,G که C ورودی فعلی

است و D,E,F,G نیز پیش‌بینی‌های زمان آینده هستند. همان‌گونه که مشاهده می‌کنید، بااینکه ورودی شبکه از B به‌صورت کامل عوض شده است و به C تبدیل شده است ولی پیش‌بینی شبکه فقط ۲۰ درصد تغییر کرده است. این امر باعث می‌شود که شبکه در خروجی ناحیه‌ها دارای پایداری خوبی باشد.

ما از اصطلاح متمرکز کننده زمانی[۱] برای توصیف دو مرحله شامل اضافه کردن مفهوم به بازنمایی و پیش‌بینی استفاده کرده‌ایم. با مدل کردن یک تغییر آرام در خروجی برای الگوهای متوالی، به‌صورت ذاتی الگوهای متفاوتی را با هم متمرکز می‌کنیم که این الگوها به دنبال هم در زمان می‌آیند.

حال به سطح دیگری از جزییات می‌رویم. ما در ابتدا با مفاهیمی از متمرکز کننده‌های زمانی و مکانی شروع می‌کنیم. سپس در مورد جزییات خاص برای هرکدام از موارد بالا نکاتی را خواهیم گفت.

۳.۳. مفاهیم اشتراکی

یادگیری در متمرکز کننده زمانی و مکانی مشابه یکدیگر است. یادگیری در هر دو مورد شامل برقراری اتصالات یا سیناپس‌ها بین سلول‌هاست. متمرکز کننده زمانی یاد می‌گیرد که اتصالات را بین سلول‌های داخل یک ناحیه برقرار کند. ولی متمرکز کننده مکانی یاد می‌گیرد که اتصالات پیش‌خور بین بیت‌های ورودی و ستون‌ها را برقرار کند.

- **وزن‌های باینری**

برخلاف خیلی از شبکه‌های عصبی که مقادیر وزن‌ها به‌صورت عددی است در این شبکه مقادیر فقط به‌صورت صفر یا یک می‌باشند.

- **پایداری**

سیناپس‌ها در طول یادگیری به‌صورت ثابت تشکیل و یا نابود می‌شوند. همان‌طور که قبلاً گفته‌شده مقدار پایداری هر سیناپس که در فاصله صفر تا یک می‌تواند باشد، میزان پایداری آن اتصال را نشان می‌دهد. برای اینکه تشخیص دهیم که یک اتصال چقدر می‌تواند تأثیر داشته باشد اگر مقدار پایداری آن از ۰.۲ بیشتر باشد می‌توان گفت که سیناپس مطرح شده به وجود می‌آید و اگر مقدار پایداری پایین‌تر از حد آستانه‌ای باشد آن سیناپس تأثیری نخواهد داشت.

- **قسمت‌های دندریتی**

سیناپس‌ها به قسمت‌های دندریتی متصل می‌شوند. دو مدل از دندریت‌ها وجود دارند. پروگزیمال و دیستال. بخش دندریت پروگزیمال سیناپس‌هایی را با ورودی‌های پیش‌خور شکل می‌دهد. سیناپس‌های فعال در این مدل، به‌صورت خطی جمع می‌شوند برای اینکه مقدار فعال‌سازی یک ستون پیش‌خور را مشخص کنند.

یک بخش دندریتی دیستال سیناپس‌هایی را شکل می‌دهد که با سلول‌هایی در یک ناحیه در ارتباط باشند. هر سلول تعداد زیادی دندریت دیستال دارد. اگر مجموع سیناپس‌های یک بخش دیستال از یک حد آستانه بالاتر برود، درنتیجه سلول‌های وابسته در یک حالت پیش‌بینی به‌صورت فعال درمی‌آیند. به خاطر اینکه برای هر سلول تعداد زیادی از بخش‌های دندریتی دیستال وجود دارد، یک حالت پیشینی برای سلول، یک عمل or منطقی است از تعدادی آشکارساز حد آستانه اصلی است.

[۱] temporal pooler

- **سیناپس‌های فعال**

همان‌گونه که قبلاً گفته شد، هر بخش دندریت دارای یک لیست از سیناپس‌های فعال است. تمام این سیناپس‌های فعال دارای یک مقدار پایداری هستند و شاید یک سیناپس اگر مقدار پایداری آن از یک حد آستانه‌ای بالاتر برود فعال بشود.

- **یادگیری**

یادگیری شامل افزایش و یا کاهش مقدار پایداری سیناپس‌های فعال در یک بخش دندریتی است. این قسمت مشابه قوانین هب[1] است. به‌عنوان‌مثال اگر یک سلول پس سیناپسی با یک بخش دندریتی که ورودی را دریافت کرده است، فعال شده باشد و در ضمن ورودی بالاتر از یک حد آستانه باشد، درنتیجه مقدار پایداری آن اتصال سیناپسی اصلاح می‌شود. پس درنتیجه سیناپس‌هایی که فعال هستند و با سیناپس‌هایی که در حال فعال شدن هستند مشارکت می‌کنند، مقدار پایداری آن‌ها زیاد می‌شود و سیناپس‌هایی که فعال نیستند و درنتیجه هیچ مشارکتی ندارند، مقدار پایداری آن‌ها کم می‌شود [2].

۴.۳. مفاهیم متمرکز کننده مکانی

مهم‌ترین تابع اساسی متمرکز کننده مکانی تبدیل ورودی‌های یک ناحیه به یک الگوی تنک است. این تابع بسیار مهم است زیرا مقدمه یادگیری و پیش‌بینی، داشتن توزیع تنک شده است. تعداد زیادی اهداف هم‌پوشانی برای متمرکز کننده مکانی وجود دارد که مشخص می‌کند متمرکز کننده مکانی تا چه حد یاد گرفته است و یا فعالیت داشته است.

۱. استفاده از تمام ستون‌ها

یک ناحیه از حافظه زمانی سلسله‌مراتبی دارای ستون‌های ثابتی است که وظیفه‌ی بازنمایی را بر عهده دارند. یک هدف برای این ناحیه این است که اطمینان حاصل پیدا کند تمام ستون‌های این ناحیه، بازنمایی چیزهای مفید را یاد گرفته است. چون ما در یک ناحیه ستون‌های غیرفعال نمی‌خواهیم. ستون‌ها با توجه به همسایگی خود فعالیت می‌کنند به‌گونه‌ای که اگر یک ستون فعالیت کمی داشته باشد، سعی می‌کند سطح فعالیت خودش را بالا ببرد و از طرفی به علت ارتباط با همسایگان و بحث رقابتی که در این همسایگی‌ها وجود دارد باعث می‌شود این ستون کم فعالیت، به تحرک بیشتر دست بزند تا بتواند در آن همسایگی یک ستون برنده باشد و درنتیجه باعث تغییر در وضعیت سایر همسایگان برای بازنمایی الگوی ورودی می‌شود.

۲. نگه‌داشتن چگالی مطلوب

در این شبکه و در یک ناحیه بین ستون‌های آن یک رقابت وجود دارد به‌گونه‌ای ستون‌ها تمایل دارند که در بازنمایی تنک توزیع‌شده برای یک ورودی برنده بشوند؛ اما هر ستون برای خود یک میدان بازدارندگی دارد که باعث می‌شود در یک شعاعی از برنده شدن دیگر ستون‌ها ممانعت کند. این شعاع می‌تواند از یک ناحیه کوچک تا سایز فضای ناحیه باشد. پس در آخر، ما فقط یک ناحیه خاص با تعدادی کمی ستون برنده داریم و بقیه ستون‌ها به حالت غیرفعال می‌روند.

۳. جلوگیری از الگوهای بدیهی و جزئی

شبکه ما باید از بازنمایی الگوهایی که ارزش بازنمایی ندارند و یا جزئی هستند، جلوگیری کند. برای همین یک حد آستانه برای فعال شدن ستون قرار می‌دهیم. به‌عنوان‌مثال عدد ۵۰ را اگر به‌عنوان حد آستانه قرار دهیم، این ستون زمانی فعال می‌شود که حداقل ۵۰ سیناپس متصل به ورودی‌های دندریتی آن فعال باشند. این مکانیزم شبیه

[1] Hebbian Rule

نورون‌های مغزی است که نورون پس سیناپسی زمانی فعال می‌شود که جمع خطی و یا غیرخطی ورودی‌هایش از یک آستانه‌ای بالاتر برود.

۴. جلوگیری از اتصالات اضافی

حافظه زمانی سلسله‌مراتبی در حین یادگیری ممکن است اتصالات سیناپسی زیادی را به وجود بیاورد (خاصیت حفظ‌کنندگی یا برازش) که باعث می‌شود خیلی الگوهای نامربوط را بازنمایی کند. به‌عنوان‌مثال نویزها. به همین دلیل برای حل این مشکل، مقدار پایداری هر سیناپس را برای اتصالاتی که در فعال شدن ستون‌های فعال نقشی ندارند و به آن متصل نیستند را کاهش می‌دهیم. این کار باعث می‌شود که هر ستون فقط یک سری الگوهای محدود و یا حتی فقط یک الگو را بیشتر بازنمایی نکند.

۵. محدوده پذیرش خود اصلاحی

مغز انعطاف‌پذیری زیادی دارد به هنگام تغییر در ورودی‌ها و یا در حین مکانیزم یادگیری و فعالیت واحدها و یا هنگامی‌که یک نورون دچار مشکل می‌شود بقیه نورون‌ها وارد عمل می‌شوند تا الگویی که قرار بود توسط این نورون پیاده شود را بازنمایی کنند. همچنین درصورتی‌که حس‌گرهای ورودی دچار مشکل شوند، نورون‌ها سعی در تخمین ورودی بر اساس یادگیری‌هایی که داشتند را انجام بدهند و با این کار یک سیستم خود اصلاح[1] و انعطاف‌پذیر را خواهیم داشت.

ما می‌خواهیم که ناحیه‌های شبکه ما نیز چنین انعطاف‌پذیری را داشته باشند که اگر به یک ناحیه ۱۰ هزار ستون اختصاص داده شده باشد، ناحیه باید یاد بگیرد که با این تعداد ستون کار خود را انجام دهد و اگر تعداد این ستون‌ها را ۲۰ هزار کنیم ناحیه باید چنین انعطافی را داشته باشد که بتواند با این تعداد کار خود را انجام دهد؛ و این ناحیه باید قادر باشد که اگر ستون‌های بیشتری به آن اختصاص داده شده است، اطلاعات بیشتری را هم با جزییات بیشتر نگه‌داری کند. البته چون در هر مرحله اکثر ستون‌ها غیرفعال هستند، تضمین تنک بودن ستون‌های فعال وجود دارد.

پس مواردی که باعث می‌شود کارکرد یک ناحیه مطلوب باشد را در این قسمت مشاهده کردیم. به‌عنوان‌مثال کاهش پایداری یک اتصال در صورت مشارکت نکردن در ستون‌های فعال، بازدارندگی همسایه‌ها و وجود سیستم رقابتی برای تضمین تنک بودن ستون‌های فعال، قرار دادن حد آستانه برای فعال شدن ستون‌ها به جهت بازنمایی نکردن موارد جزئی و بی‌ارزش، خود تطبیقی تعداد سیناپس‌ها و کم‌وزیاد کردن آن‌ها که همه این‌ها باعث یک سیستم جمعی پویا برای بازنمایی و کارکرد این شبکه هستند [2].

۵.۳. جزییات متمرکز کننده مکانی

حالا ما می‌توانیم مشخص کنیم که در متمرکز کننده مکانی چه فعالیت‌هایی انجام می‌شود که به شرح زیر است.

۱. شروع با یک الگوی ورودی با یک تعداد بیت ثابت. ورودی یا از حسگر یا از نواحی دیگر می‌آید.
۲. تعدادی ستون ثابت را به یک ناحیه اختصاص می‌دهیم که با استفاده از نواحی دندریتی و اتصالات این کار انجام می‌شود. هر قسمت دندریت یک مجموعه‌ای از سیناپس‌های فعال است که قسمتی از الگوی ورودی را بازنمایی می‌کند. هر سیناپس یک مقدار پایداری دارد که این مقدار معتبر بودن سیناپس‌های فعال را مشخص می‌کند
۳. شمارش تعداد سیناپس‌های معتبر در یک ستون که به یک ورودی فعال متصل هستند

[1] Self adjusting

۴. محاسبه فاکتور تقویت[۱] برای تنظیم مقادیر پایداری بر اساس میزان و تعداد دفعات فعال بودن یک ستون نسبت به همسایگانی که دارد.
۵. ستونی که بالاترین فعالیت را دارد باعث می‌شود که در یک محدوده‌ای، همسایگانش را غیرفعال کند. این باعث می‌شود که تعداد زیادی از ستون‌ها غیرفعال شوند و یک بازنمایی تنک به وجود آید. این ناحیه که برای غیرفعال کردن همسایگان است به‌صورت پویا بر اساس گستردگی (تعداد ورودی) هایی از بیت‌های ورودی است.
۶. و در مرحله آخر مقادیر پایداری سیناپس‌ها را اصلاح می‌کنیم به این صورت که برای ستون فعال، سیناپس‌هایی که به ورودی‌های فعال متصل هستند مقدار پایداری را افزایش می‌دهیم و برای سیناپس‌ها با ورودی‌های غیرفعال، این مقدار را کاهش می‌دهیم درنتیجه این کار ممکن است بعضی از سیناپس‌های غیر معتبر را معتبر کند و بعضی را نیز برعکس [2].

۶.۳. مفاهیم متمرکز کننده زمانی

اگر به یاد بیاورید در قسمت‌های قبلی گفتیم که متمرکز کننده زمانی باعث حفظ دنباله‌ها و انجام پیش‌بینی می‌شود. پایه این کار به این صورت است که وقتی یک سلول فعال می‌شود تعدادی اتصال با سلول‌هایی که از قبل (در زمان نزدیک قبلی) فعال بودند می‌سازد. پیش‌بینی‌هایی که این سلول انجام می‌دهد بر اساس همین ارتباطات است. اگر در یک ناحیه تمام سلول‌ها این کار را انجام دهند ما می‌توانیم در آن ناحیه، یک سری دنباله را ذخیره و یا اتفاقاتی را تخمین بزنیم.

چون سیستم ذخیره اطلاعات به‌صورت توزیع‌شده است، دنباله‌های ذخیره‌شده جای خاصی در حافظه وجود ندارند. این باعث می‌شود که سیستم در مقابل خرابی و نویز مقاوم باشد؛ و حتی در صورت خرابی بعضی از قسمت‌ها، بازهم سیستم به خوبی می‌تواند اطلاعات را بازیابی کند و به کار خود با کمترین خطای ممکن ادامه دهد. چیزی مشابه با خرابی آبرومندانه[۲] در سیستم‌های هوش محاسباتی[۳].

به‌عنوان‌مثال فرض کنید که یک ناحیه شامل ۱۰ هزار ستون داریم که در هرلحظه فقط ۲۰۰ سلول می‌توانند فعال باشند. ما چگونه یک الگوی خاص را با استفاده از این ۲۰۰ سلول ذخیره می‌کنیم؟ یک کار ساده به این صورت است که یک لیستی تهیه کنیم و بگوییم زمانی که این ۲۰۰ سلول فعال هستند الگوی خاصی مد نظر است. حال اگر بگوییم فقط ۲۰ سلول از این ۲۰۰ تا را نگه‌داری کنیم چه می‌شود؟ آیا بازهم سیستم این الگو را به یاد می‌آورد و یا اینکه دچار مشکل می‌شود؟ به علت اینکه حافظه ما توزیع‌شده است و از طرفی به علت بزرگی هر ناحیه (۱۰ هزار ستون در هر ناحیه) ما می‌توانیم با خطای کمی با همین ۲۰ سلول که برای یک الگو ذخیره کردیم، نیز الگو را به یاد بیاوریم. این توزیع‌شدگی باعث می‌شود که برای الگوهای متفاوت، تعداد ۲۰ ستون خاصی که قرار است ذخیره کنیم هم‌پوشانی و یا تشابه کمی داشته باشند؛ و مزیت دیگر این است که حافظه مورد استفاده بسیار کاهش پیدا کند.

هر سلول به‌طورمعمول تعداد زیادی دنباله و الگو را تداعی می‌کند. یک سری سلول‌های عمومی هستند که در اکثر الگوها مشارکت دارند. درنتیجه برای هر سلول ممکن است تعداد زیادی قسمت دندریتی فعال یا غیرفعال وجود داشته باشد. ایده آل این است که یک قسمت دندریتی برای هر الگو فعال باشد و آن را یادآوری کند ولی بااین‌وجود

---

[۱] boosting factor

[۲] Graceful Degradation

[۳] Computational Intelligence

اینکه تعداد زیادی قسمت دندریتی داریم ولی بازهم سیستم به خوبی کار می‌کند و این اتصالات را به خوبی برای الگوهای متفاوت پیچیده یاد می‌گیرد.

به‌عنوان‌مثال یک ناحیه برای ۴ الگوی متفاوت که برای هرکدام ۲۰ اتصال فعال را ذخیره می‌کند در نظر بگیرد. این ناحیه با فعال شدن این ۲۰ عدد اتصال، هرکدام از الگوهای ذخیره‌شده را یادآوری کند؛ یعنی در کل ۴ قسمت دندریتی برای این ناحیه داریم. حال اگر یک حد آستانه ۱۵ تایی برای فعال شدن هر قسمت دندریتی بگذاریم و بگوییم قسمت دندریتی زمانی فعال شود که ۱۵ اتصال به اتصالات فعال داشته باشد. در این قسمت یک احتمال خطایی معرفی می‌شود مبنی بر اینکه اگر این ۱۵ اتصال مخلوطی از ۴ الگوی بالا باشد، سیستم دچار خطا می‌شود و نمی‌داند که کدام الگو را بازیابی کند. ولی این احتمال بسیار کم است چون همان‌طور که قبلاً هم گفتیم به علت توزیع تنک ستون‌های فعال، این احتمال که این ۴ الگو در ۲۰ اتصال فعال، اشتراکات زیادی داشته باشند بسیار کم است.

در ادامه به توصیف یک سلول با یک یا چند قسمت دندریتی و تعداد زیادی اتصالات سیناپسی می‌پردازیم که می‌تواند صدها حالت مختلف از سلول‌های فعال و اتصالاتی که به این سلول هست را شناسایی کند و الگوی خاصی که تداعی می‌شود را بازنمایی کند [2].

۷.۳. جزییات متمرکز کننده زمانی

در این قسمت مراحل کاری متمرکز کننده زمانی را توضیح می‌دهیم. شروع کار این قسمت، بعد از تمام شدن کار متمرکز کننده مکانی است که در پایان کار آن یک مجموعه فعال از ستون‌ها را که نماینده ورودی پیش‌خور است به ما می‌دهد.

۱. برای هر ستون فعال، به ازای هر سلولی که در آن ستون در حالت تخمین باشد، این سلول‌ها را فعال می‌کنیم. درصورتی‌که هیچ سلولی در حالت تخمین قرار نداشت همه سلول‌ها را فعال می‌کنیم. این مجموعه سلول فعال شده معادل است با بازنمایی ورودی در مفهوم ورودی قبلی.

۲. برای هر قسمت دندریتی در سلول‌های فعال تعداد سیناپس‌هایی که به سلول‌های فعال دیگر متصل هست را می‌شماریم. اگر این تعداد بیشتر از حد آستانه بود، این بخش دندریتی فعال می‌شود. سپس سلول‌های با بخش دندریتی فعال به حالت پیش‌بینی می‌روند مگر اینکه قبلاً توسط یک ورودی پیش‌خور فعال شده باشند. سلول‌هایی که بخش دندریتی فعال ندارند و یا قبلاً توسط یک ورودی پایین به بالا (از سطوح دیگر شبکه) فعال نشده بودند به حالت غیرفعال می‌روند و اگر غیرفعال بودند در همان حالت می‌مانند.

۳. زمانی که یک قسمت دندریتی فعال شد، برای اتصالات سیناپسی آن‌که به سلول‌های فعال متصل است مقدار پایداری را افزایش می‌دهیم و برای اتصالات سیناپسی که به سلول‌های غیرفعال متصل هستند این مقدار پایداری را کاهش می‌دهیم. این تغییرات یک برچسب موقتی هستند که در مراحل بعد اعمال و یا حذف می‌شوند.

این تغییرات باعث می‌شود که یک بخش دندریتی که قبلاً آموزش دیده شده است به حدی کافی برسد و فعال شود و به حالت تخمین برود. بااین‌وجود ما می‌خواهیم که این بخش دندریتی بر اساس گذشته دورتر تخمین بزند به همین خاطر یک بخش دندریتی دومی را در همان سلول انتخاب می‌کنیم. این بخش باید به‌گونه‌ای انتخاب شود که نزدیک‌ترین حالت به حالت قبلی سیستم باشد. سپس برای این بخش نیز تغییر مقادیر پایداری را برای اتصالات به آن، اصلاح می‌کنیم. تغییرات انجام شده برای این بخش دندریتی هم برچسب موقتی خواهد داشت.

۴. هر زمان که یک سلول به علت ورودی پیش‌خور، از حالت غیرفعال به حالت فعال تغییر کند درنتیجه سیناپس‌های بالقوه‌ای که به سلول فعال شده، وابسته هستند و برچسب‌های موقتی که به آن‌ها داده شده بود را حذف می‌کنیم؛ بنابراین ما مقدار پایداری سیناپس سلولی را اصلاح می‌کنیم که به‌صورت کاملاً صحیح مقدار فعالیت پیش‌خور آن سلول را پیش‌بینی کرده باشد.
۵. زمانی که یک سلول از حالت فعال به حالت غیرفعال می‌رود برای هر سیناپس فعال در این سلول تمام علائم موقتی را که قبلاً داده بودیم به مقدار قبلی بازمی‌گردانیم. چون ما هیچ‌وقت نمی‌خواهیم اتصالات سیناپسی که پیش‌بینی اشتباهی داشته‌اند، تقویت بشوند.

توجه داشته باشید که تنها سلول‌هایی که به علت ورودی‌های پیش‌خور فعال شده بودند، باعث فعال شدن سلول‌های دیگر در آن ناحیه می‌شوند. در غیر این صورت پیش‌بینی‌ها به سمت پیش‌بینی‌های دورتر می‌رفت؛ اما تمام سلول‌های فعال (چه به علت ورودی پیش‌خور و چه در حالت پیش‌بینی) خروجی یک ناحیه را تشکیل می‌دهند و این خروجی را به نواحی بعدی در سلسله‌مراتب شبکه انتشار می‌دهند [2].

۸.۳. دنباله‌های مرتبه اول و مرتبه‌های متغیر و پیش‌بینی

یک موضوع مهم دیگر نیز وجود دارد که البته این موضوع برای فهمیدن ادامه مطالب این گزارش نیاز نیست ولی یکی از قابلیت‌های شبکه حافظه زمانی سلسله‌مراتبی است. هم‌اکنون این سوال پیش می‌آید که چه اتفاقی می‌افتد که اگر تعداد سلول‌های موجود در یک ستون را کم و یا زیاد کنیم؟ و یا اینکه در هر ستون فقط یک سلول وجود داشته باشد؟

به‌عنوان‌مثال در آزمایش‌هایی که قبلاً انجام شده بود، یک شبکه دارای ۱۰۰ ستون فعال که در هر ستون ۴ سلول بود را پیاده کرده‌اند. تعداد حالاتی که این ناحیه می‌تواند به ما بدهد حدود 4^{100} است. حتی ممکن است که یک ورودی مشابه حالات بازنمایی مختلفی را داشته باشد که با توجه به مفاهیم در زمان‌های مختلف، متفاوت خواهد بود. این شبکه این حالات مشابه را بدون هیچ سردرگمی می‌تواند تشخیص دهد.

این قابلیت به این صورت است که به‌عنوان‌مثال یک ناحیه می‌تواند یک جمله، با کلمات مشابه را که بارها و بارها تکرار شده است را به یاد آورد. به این مکانیزم اصطلاحاً پیش‌بینی مرتبه متغیر[1] میگویند؛ یعنی پیش‌بینی‌های انجام شده فقط بر اساس چیزی که الآن رخ داده است به وجود نمی‌آید بلکه بر اساس مفاهیم گذشته نیز این پیش‌بینی تحت تأثیر قرار خواهد گرفت. پس حافظه زمانی سلسله‌مراتبی یک حافظه مرتبه متغیر است.

حال اگر تعداد سلول‌های در هر ستون را به عدد ۵ تغییر دهیم تعداد حالاتی که می‌توانیم از ورودی‌ها داشته باشیم به عدد 5^{100} خواهد رسید؛ که این تعداد نسبت به حالت قبلی که ۴ سلول در هر ستون بود بسیار بیشتر خواهد شد ولی آیا این مقدار بالا واقعاً نیاز خواهد بود؟

بااین‌حال، تعداد سلول در ستون را کمتر کنیم باعث به وجود آمدن اختلافات بیشتر خواهد شد. به‌عنوان‌مثال اگر ما تعداد سلول در هر ستون را به عدد یک کاهش دهیم، این احتمال و توانایی از شبکه گرفته می‌شود که بتواند یک الگو را در مفاهیم متعدد بازنمایی‌هایی متعدد از آن داشته باشد. یک ورودی همیشه یک پیش‌بینی مشابه را بر اساس فعالیت قبلی که بوده است به ما می‌دهد. اگر چنین باشد که در هر ستون فقط یک سلول موجود باشد، حافظه زمانی سلسله‌مراتبی یک حافظه مرتبه اول[2] خواهد بود و پیش‌بینی‌ها فقط بر اساس ورودی فعلی است.

---

[1] variable order

[2] first order

این حافظه مرتبه اول برای مسائلی خوب است که زمان و تغییرات در آن نباشد؛ اما در شبکه حافظه زمانی سلسله‌مراتبی، هم تشخیص بر حسب دنباله‌های زمانی را خواهیم داشت که همان سیستم چند سلول در هر ستون است و هم تشخیص محیط‌های استاتیک که می‌تواند توسط ناحیه‌هایی انجام شود که در هر ستون آن‌ها یک سلول است.

نواحی حافظه زمانی سلسله‌مراتبی سعی می‌کنند که جزییات عمومی یک الگو را بیشتر در نظر داشته باشند (به‌عنوان‌مثال حاشیه‌های تصاویر) که باعث می‌شود این شبکه در صورت تغییر این ورودی، این جزییات عمومی را تعقیب کند. به‌عنوان‌مثال اگر یک ناحیه با ستون‌های تک‌سلولی را به یک خط افقی بدهیم که در طی زمان به سمت راست حرکت می‌کند چه اتفاقی می‌افتد؟ در این صورت ناحیه ما فقط مشخص می‌کند که خط به کدام سمت حرکت می‌کند، ولی اینکه از مفهوم‌های قبلی که خط قبلاً در کجا بوده است بهره‌ای نخواهد برد. این مدل شبیه سلول‌های پیچیده[1] در مغز می‌باشند؛ اما زمانی که حافظه زمانی سلسله‌مراتبی از نواحی با ستون‌های چند سلوله استفاده کند شبیه به سلول‌های پیچیده تنظیم شده هدایتی[2] در مغز می‌باشند. در این مدل پیش‌بینی به این صورت خواهد بود که خط یا به سمت راست حرکت خواهد کرد و یا سمت چپ و نه پیش‌بینی هر دو. ولی در مدل قبلی پیشی بینی به این صورت خواهد بود که خط یا به راست یا چپ و یا اصلاً حرکت نخواهد کرد.

به‌صورت کلی می‌توان گفت که شبکه باید هم از حافظه مرتبه اول و هم مرتبه متغیر استفاده کند. هر ناحیه می‌تواند شامل ۴ یا ۵ لایه سلول‌ها باشد. هر لایه از جهاتی با هم متفاوت ولی به‌صورت ستونی با هم در ارتباط هستند و اتصالات زیاد افقی در خود لایه نیز وجود دارد. هر لایه از سلول‌ها در این شبکه نقش جداگانه‌ای از سایر لایه‌ها خواهند داشت. به‌عنوان‌مثال از علم تشریح بدن چنین به دست آمده است که لایه ششم وظیفه بازخورد در سلسله‌مراتب را دارد و لایه پنجم در مباحث حرکات رفتاری مشارکت می‌کند؛ و دو لایه اصلی که به‌صورت پیش‌خور عمل می‌کنند لایه‌های سوم و چهارم هستند به نظر ما چنین است که تفاوت میان لایه سوم و چهارم در این است که در لایه چهارم سلول‌های به‌صورت مستقل‌تری نسبت لایه سوم کار می‌کنند، شبیه ستون‌هایی با یک سلول. ولی در لایه سوم ارتباطات بیشتر و ستون‌های چند سلولی حضور دراند. لایه‌های اول و دوم هم به دو صورت مرتبه اول و متغیر پیاده‌سازی می‌شوند که لایه مرتبه اول (شبیه لایه چهارم) بیشتر روی مباحث بازنمایی الگوهایی که به‌صورت مکانی تغییرات ندارند و لایه مرتبه متغیر (مشابه لایه سوم) روی یادگیری دنباله‌های زمانی و پیش‌بینی عمل می‌کند.

به‌صورت خلاصه در این شبکه سعی شده است که رفتارهایی که در قشر مغزی وجود دارد تا حد امکان به‌صورت ساده‌ای پیاده‌سازی کند؛ اما لایه‌های مغزی دارای جزییات فراوان و قواعد پیچیده‌تری هستند که باعث می‌شود نقش‌های متفاوت‌تری را نسبت به بازخورد، پیش‌خور یا توجه و حرکات رفتاری داشته باشد [2].

۴. پیاده‌سازی متمرکز کننده مکانی و شبه کد آن

در این فصل قصد داریم که در مورد الگوریتم متمرکز کننده مکانی صحبت کنیم. ورودی این الگوریتم یک آرایه از ورودی‌های پایین به بالا است که از حس‌گرهای اطلاعاتی و یا از لایه‌های قبلی آمده است؛ که در این روال ستون‌های فعال ActiveColumns(t) شمرده می‌شوند و سپس این ActiveColumns(t) به متمرکز کننده زمانی که در فصل بعدی توضیح داده می‌شود فرستاده می‌شود. پس ActiveColumns(t) به‌عنوان خروجی متمرکز کننده مکانی حساب می‌شود.

---

[1] complex cells

[2] directionally-tuned complex cells

این قسمت از الگوریتم شامل سه فاز است:

۱. محاسبه اشتراک[۱] به ازای هر ستون با استفاده از ورودی
۲. محاسبه ستون‌های فعال برنده بعد از بازنمایی ورودی
۳. به‌روزرسانی مقادیر پایداری سیناپس‌ها و متغیرهای داخلی

این الگوریتم خاصیت یادگیری برخط دارد به‌گونه‌ای که برای قطع یادگیری می‌توان فاز سوم این الگوریتم را انجام نداد [2].

۱.۴. مقداری دهی اولیه

قبل از اینکه ورودی توسط یک ناحیه دریافت شود، مقداردهی اولیه این ناحیه با محاسبه یک لیست از سیناپس‌های فعال برای هر ستون انجام می‌شود. این لیست شامل یک مجموعه تصادفی از ورودی‌هاست که از فضای ورودی مسئله انتخاب شده‌اند. هر ورودی توسط یک سیناپس و یک مقدار پایداری مخصوص، نمایش داده می‌شود. مقدار پایداری برای این سیناپس با دو شرط انتخاب می‌شود. اول اینکه این مقدار باید کوچک و نزدیک مقدار ConnectedPerm باشد. این متغیر حد آستانه‌ای است که فعال بودن این سیناپس را نشان می‌دهد. این نوع مقداردهی باعث می‌شود که این سیناپس بعد از چند تکرار کوتاه آموزش، فعال بودن یا غیرفعال بودنش کاملاً مشخص شود. دوم اینکه هر ستون یک مرکز طبیعی[۲] بیش از ورودی دارد و مقادیر پایداری در ستون، یک بایاس نسبت به این مرکز دارند. آن‌هایی که نزدیک‌تر به این مرکز هستند بایاس بیشتری دارند [2].

۲.۴. فاز اول: اشتراک (همپوشانی)

فرض کنید که یک بردار ورودی به ناحیه وارد می‌شود. در این فاز به محاسبه مقدار همپوشانی ستون‌ها با بردار ورودی فعلی می‌پردازیم. این کار به‌سادگی انجام می‌شود به‌گونه‌ای که تعداد سیناپس‌های متصل به ورودی فعال در یک ستون را محاسبه می‌کنیم و در مقدار تقویت[۳] این ستون ضرب می‌کنیم [2].

```
1.  for c in columns
2.
3.          overlap(c) = 0
4.          for s in connectedSynapses(c)
5.                  overlap(c) = overlap(c) + input(t, s.sourceInput)
6.
7.          if overlap(c) < minOverlap then
8.                  overlap(c) = 0
9.          else
10.         overlap(c) = overlap(c) * boost(c)
```

شکل ۸ – شبه کد فاز اول متمرکز کننده مکانی [2]

۳.۴. فاز دوم: بازدارندگی

در این فاز به تعیین ستون‌های برنده می‌پردازیم. در این قسمت حد آستانه‌ای (desiredLocalActivity) را مشخص می‌کنیم که بر اساس ستون‌های برنده مشخص می‌شوند. به‌عنوان‌مثال اگر مقدار این متغیر برابر با ۱۰ باشد،

---

[۱] Overlap

[۲] natural center

[۳] Boost

زمانی یک ستون برنده خواهد شد که مقدار همپوشانی آن‌که در فاز اول محاسبه شده بود بزرگ‌تر از مقدار همپوشانی دهمین بزرگ‌ترین ستون در میدان بازدارندگی خود باشد [2].

```
11. for c in columns
12.
13.     minLocalActivity = kthScore(neighbors(c), desiredLocalActivity)
14.
15.     if overlap(c) > 0 and overlap(c) ≥ minLocalActivity then
16.         activeColumns(t).append(c)
17.
```

شکل ۹ – شبه کد فاز دوم متمرکز کننده مکانی [2]

۴.۴. فاز سوم: یادگیری

قسمت سوم فاز یادگیری است که در این فاز به به‌روزرسانی مقادیر پایداری هر سیناپس در صورت لزوم و همچنین به‌روزرسانی مقادیر تقویت و محدوده بازدارندگی می‌پردازیم. قسمت اصلی این فاز در خطوط ۲۰ تا ۲۶ است جایی که اگر اتصال سیناپسی فعال بود مقدار پایداری این اتصال افزایش میابد و در غیر این صورت مقدار آن کم خواهد شد. مقادیر پایداری اتصالات سیناپسی باید بین مقادیر ۰ تا ۱ باشند.

در خطوط ۲۸ تا ۳۰ هم بحث تقویت پیاده‌سازی شده است. در اینجا دو مدل مکانیزم تقویت را خواهیم داشت که به ستون‌ها کمک می‌کند تا اتصالات را یاد بگیرند. اگر یک ستون به اندازه کافی برنده نشود (با مقدار activeDutyCycle مشخص می‌شود) درنتیجه مقدار تقویت آن به‌طورکلی افزایش می‌یابد. (خطوط ۳۰ تا ۳۲)

متعاقباً اگر اتصالات سیناپسی یک ستون به هیچ یک از ورودی‌ها همپوشانی و تطابق نداشت (با متغیر overlapDutyCycle تخمین زده می‌شوند)، مقدار پایداری این سیناپس افزایش می‌یابد (خطوط ۳۴ تا ۳۶). توجه شود که اگر فاز یادگیری خاموش باشد، مقدار افزایش برای هر ستون ثابت خواهد ماند؛ و در آخر و در خط ۳۸ میدان بازدارندگی برای هر ستون محاسبه می‌گردد [2]. توضیحات مقادیر و توابع استفاده شده در پیاده‌سازی متمرکز کننده مکانی در قسمت پیوست‌ها آمده است.

```
18. for c in activeColumns(t)
19.
20.     for s in potentialSynapses(c)
21.         if active(s) then
22.             s.permanence += permanenceInc
23.             s.permanence = min(1.0, s.permanence)
24.         else
25.             s.permanence -= permanenceDec
26.             s.permanence = max(0.0, s.permanence)
27.
28. for c in columns:
29.
30.     minDutyCycle(c) = 0.01 * maxDutyCycle(neighbors(c))
31.     activeDutyCycle(c) = updateActiveDutyCycle(c)
32.     boost(c) = boostFunction(activeDutyCycle(c), minDutyCycle(c))
33.
34.     overlapDutyCycle(c) = updateOverlapDutyCycle(c)
35.     if overlapDutyCycle(c) < minDutyCycle(c) then
36.         increasePermanences(c, 0.1*connectedPerm)
37.
38. inhibitionRadius = averageReceptiveFieldSize()
39.
```

شکل ۱۰ – فاز یادگیری در متمرکز کننده مکانی [2]

۵. پیاده‌سازی متمرکز کننده زمانی و شبه کد آن

در این فصل به بررسی شبه کد برای متمرکز کننده زمانی می‌پردازیم جایی که ورودی آن activeColumns در زمان t است که خروجی متمرکز کننده مکانی بود. در این قسمت حالت پیش‌بینی و فعال برای هر سلول در زمان فعلی t محاسبه می‌کنیم. خروجی OR دو متغیر پیش‌بینی و فعال بودن برای هر سلول، خروجی متمرکز کننده زمانی را می‌دهد که برای مرحله بعدی استفاده می‌شود.

این بخش نیز به سه فاز جداگانه تقسیم می‌شود که این سه فاز به شرح زیر می‌باشند:

۱. فاز محاسبه حالات فعال activeState(t) برای هر سلول
۲. فاز محاسبه کردن حالت پیش‌بینی predictiveState(t) برای هر سلول
۳. به‌روزرسانی سیناپس‌ها

باید توجه کرد که فاز سوم تنها برای بخش یادگیری است و درصورتی‌که نیازی به یادگیری نباشد این قسمت را انجام نمی‌دهیم. هرچند، برخلاف متمرکز کننده مکانی، در فازهای اول و دوم متمرکز کننده زمانی یک سری عملگرهای خاص یادگیری وجود دارد.

متمرکز کننده زمانی از ساختار پیچیده‌تری نسبت به متمرکز کننده مکانی برخوردار است، به همین دلیل در ابتدا فقط نسخه استنتاج متمرکز کننده زمانی را لیست کرده‌ایم و سپس به دنبال آن مدل استنتاج و یادگیری آن را شرح می‌دهیم. توضیح بعضی از جزییات، اصطلاحات و فرآیندهای مورد استفاده در آخر فصل بعد از شبه کد قرار داده شده است [2].

۱.۵. متمرکز کننده مکانی نسخه فقط استنتاج

۱.۱.۵. فاز اول

در این فاز به محاسبه حالت فعال برای هر سلول می‌پردازیم. برای هر ستون برنده، ما تعیین می‌کنیم که کدام سلول‌ها باید فعال شوند. درصورتی‌که یک ورودی پایین به بالا توسط این ستون و هر سلولی از آن پیش‌بینی شده باشد (اگر متغیر predictiveState برابر یک باشد به علت یک قسمت دنباله‌دار در مرحله زمانی قبلی) سپس این سلول‌ها به‌صورت فعال درمی‌آیند (خطوط ۴ تا ۹). اگر ورودی پایین به بالا به‌گونه‌ای باشد که پیش‌بینی درستی انجام نداده باشد، (هیچ سلولی predictiveState روشن نشده باشد) سپس تمام سلول‌های داخل این ستون به حالت فعال درمی‌آیند [2]. (خطوط ۱۱ تا ۱۳)

```
1.  for c in activeColumns(t)
2.
3.      buPredicted = false
4.      for i = 0 to cellsPerColumn - 1
5.          if predictiveState(c, i, t-1) == true then
6.              s = getActiveSegment(c, i, t-1, activeState)
7.              if s.sequenceSegment == true then
8.                  buPredicted = true
9.                  activeState(c, i, t) = 1
10.
11.     if buPredicted == false then
12.         for i = 0 to cellsPerColumn - 1
13.             activeState(c, i, t) = 1
```

شکل ۱۱ – فاز اول متمرکز کننده زمانی فقط استنتاج [2]

۲.۱.۵. فاز دوم

در این فاز به محاسبه حالات پیش‌بینی برای هر سلول می‌پردازیم. یک سلول حالت پیش‌بینی خود را فعال می‌کند اگر حداقل یکی از قسمت‌های آن به‌صورت فعال دربیاید. به‌عنوان‌مثال اگر یک سلول به اندازه کافی اتصالاتی افقی در حالت فعال داشته باشد که این اتصالات به علت ورودی پیش‌خور فعال شده باشند [2].

```
14. for c, i in cells
15.     for s in segments(c, i)
16.         if segmentActive(c, i, s, t) then
17.             predictiveState(c, i, t) = 1
```

شکل ۱۲ – فاز دوم متمرکز کننده زمانی فقط استنتاج [2]

۲.۵. شبه کد متمرکز کننده زمانی: ترکیب یادگیری و استنتاج

در بخش قبلی متمرکز کننده زمانی که فقط قسمت استنتاج را داشت، معرفی کردیم و یادگیری برای آن در نظر نگرفتیم. این کار صرفاً برای این بود که قسمت استنتاج کمی واضح‌تر برای مخاطب جلوه کند؛ اما در این بخش از گزارش، بحث یادگیری را نیز به این متمرکز کننده اضافه کرده‌ایم که در سه فاز به شرح این روال خواهیم پرداخت.

۱.۲.۵. فاز اول

در فاز اول به محاسبه حالت فعال برای هر سلول در ستون برنده می‌پردازیم. برای این ستون‌ها، یک سلول به‌عنوان سلول حالت یادگیری انتخاب می‌شود (متغیر LearnState برای آن‌ها فعال می‌شود). منطق این کار به‌صورت زیر خواهد بود:

اگر ورودی پایین به بالا توسط سلول‌هایی پیش‌بینی شده بود، آنگاه این سلول‌ها به حالت فعال خواهند رفت خطوط ۲۳ -۲۷. اگر آن قسمت از سلول‌هایی که فعال شده بودند و مقدار Learnstate آن‌ها فعال باشد، این سلول‌ها به‌عنوان سلول یادگیری انتخاب می‌شوند. خطوط ۲۸-۳۰. ولی اگر خروجی توسط هیچ سلولی پیش‌بینی نشده بود، تمام سلول‌های این ستون به حالت فعال خواهند رفت. خطوط ۳۲ -۳۴

در ضمن اگر هیچ سلولی به‌عنوان سلول یادگیری انتخاب نشد، بهترین سلول مطابق با ورودی فعلی به‌عنوان سلول یادگیری انتخاب می‌شود خطوط ۳۶ تا ۴۱ و یک بخش جدید به این سلول اضافه می‌شود.

```
18. for c in activeColumns(t)
19.
20.     buPredicted = false
21.     lcChosen = false
22.     for i = 0 to cellsPerColumn - 1
23.         if predictiveState(c, i, t-1) == true then
24.             s = getActiveSegment(c, i, t-1, activeState)
25.             if s.sequenceSegment == true then
26.                 buPredicted = true
27.                 activeState(c, i, t) = 1
28.                 if segmentActive(s, t-1, learnState) then
29.                     lcChosen = true
30.                     learnState(c, i, t) = 1
31.
32.     if buPredicted == false then
33.         for i = 0 to cellsPerColumn - 1
34.             activeState(c, i, t) = 1
35.
36.     if lcChosen == false then
37.         l,s = getBestMatchingCell(c, t-1)
38.         learnState(c, i, t) = 1
39.         sUpdate = getSegmentActiveSynapses (c, i, s, t-1, true)
40.         sUpdate.sequenceSegment = true
41.         segmentUpdateList.add(sUpdate)
```

شکل ۱۳ – فاز اول متمرکز کننده زمانی [2]

۲.۲.۵. فاز دوم

فاز دوم برای هر سلول حالت پیش‌بینی را محاسبه می‌کند. حالت پیش‌بینی یک سلول فعال خواهد شد اگر یکی از قسمت‌های دندریتی این سلول فعال شده باشند. به‌عنوان‌مثال: اگر اتصالات جانبی آن به اندازه کافی به علت ورودی پیش‌خوری که موجود است، فعال شده باشد. در این مورد، سلول تغییرات زیر را به ترتیب انجام می‌دهد.

۱. تقویت بخش فعال فعلی خطوط ۴۷-۴۸
۲. تقویت بخش‌هایی که این فعالیت را پیش‌بینی کرده بودند به‌عنوان‌مثال بخش‌هایی که به‌صورت ضعیف توانسته بودند فعالیتی که بر اساس بازه‌های زمانی قبلی بوده است را تخمین بزنند. خطوط ۵۰ -۵۳

```
42. for c, i in cells
43.     for s in segments(c, i)
44.         if segmentActive(s, t, activeState) then
45.             predictiveState(c, i, t) = 1
46.
47.             activeUpdate = getSegmentActiveSynapses (c, i, s, t, false)
48.             segmentUpdateList.add(activeUpdate)
49.
50.             predSegment = getBestMatchingSegment(c, i, t-1)
51.             predUpdate = getSegmentActiveSynapses(
52.                                 c, i, predSegment, t-1, true)
53.             segmentUpdateList.add(predUpdate)
```

شکل ۱۴ – شبه کد فاز دوم متمرکز کننده زمانی [2]

۳.۲.۵. فاز سوم

سومین و آخر فاز وظیفه یادگیری را بر عهده دارد. در این فاز بخش‌ها، مواردی را که به خاطر ورودی پیش‌خور دریافت کرده بودند و در فاز دوم به ترتیب اصلاح شده بودند را به‌روزرسانی می‌کنند و از طرفی یک سلول به‌عنوان سلول یادگیری انتخاب می‌شود. در غیر این صورت اگر سلول به هر دلیلی پیش‌بینی را متوقف کرد ما این قسمت را تضعیف خواهیم کرد [2]. (خطوط ۵۸ – ۶۰) توضیحات اضافی و تکمیلی توابع و مقادیر مربوط به متمرکز کننده زمانی در قسمت پیوست آورده شده است.

```
54. for c, i in cells
55.     if learnState(s, i, t) == 1 then
56.         adaptSegments (segmentUpdateList(c, i), true)
57.         segmentUpdateList(c, i).delete()
58.     else if predictiveState(c, i, t) == 0 and predictiveState(c, i, t-1)==1 then
59.         adaptSegments (segmentUpdateList(c,i), false)
60.         segmentUpdateList(c, i).delete()
61.
```

شکل ۱۵- شبه کد فاز یادگیری متمرکز کننده زمانی [2]

۶. کاربردی از حافظه زمانی سلسله مراتبی

بعد از ارائه گزارش حال به کاربردی از این حافظه می‌پردازیم که در مورد زبان اشاره و نحوه شناسایی حرکات در این زبان را مورد بحث قرار می‌دهد. در این کاربرد شبکه باید قادر باشد که حرکات دست، بازو و انگشتان را دنبال کند و بعد از دنبال کردن آن بگوید که این حرکات نشان دهنده چه مفهومی خواهند بود [1].

البته این بخش، صرفاً توضیحات کلی در مورد پیاده‌سازی را داده‌ایم و از جزییات و روابط موجود در این پیاده‌سازی دوری کرده‌ایم. در مقاله‌ی [1] که ارائه شده است، از این حافظه جهت انجام شناسایی زبان اشاره استفاده شده است. برای بهبود کارایی و جلوگیری از ضعف‌هایی که حافظه زمانی سلسله مراتبی استاندارد دارد، کمی تغییرات و را در آن اعمال کرده‌اند [1].

۱.۶. مقدمه

شناسایی زبان اشاره، مقوله‌ای است روی آن زیاد کار شده است و از الگوریتم‌های و روش‌های گوناگونی برای آن استفاده شده است. به عنوان مثال مدل مخفی مارکوف و یا روش تی-کلاس[1] از روش‌هایی برای این کاربرد می‌باشند؛ اما روشی مثل تی-کلاس جنبه یادگیری با نظارت دارد. در این مقاله توانسته‌اند برای ۹۵ گروه زبان اشاره صحت ۹۱ درصدی را به دست بیاورند[1]. این مقاله روی زبان اشاره استرالیایی کار کرده است؛ زیرا زبان‌های اشاره در کشورهای مختلف تقریباً متفاوت است. به عنوان مثال زبان اشاره فارسی نیز متفاوت از زبان اشاره بقیه کشورهاست [7].

در پیاده‌سازی‌های دیگری که از این کاربرد وجود داشته است، کارهایی انجام شده است که باعث شده عملکرد آن‌ها دقت بالایی داشته باشند. به عنوان مثال پیش‌پردازش ورودی‌ها و یا استفاده از یک مدل زبان خاص برای

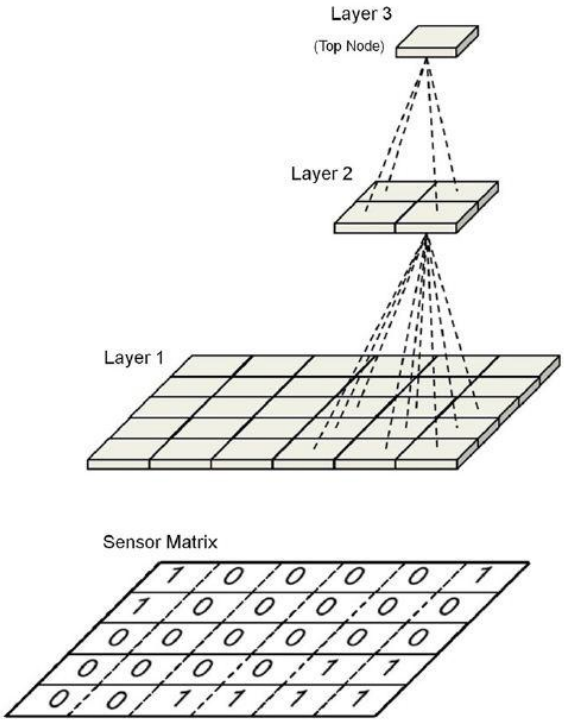
[1] T-Class

آموزش. در روش‌های دیگر، ورودی‌ها یا به صورت سیگنال الکتریکی بوده‌اند و یا به صورت ویدئویی؛ اما در شبکه حافظه زمانی سلسله مراتبی ورودی مهم نیست [1].

روش حافظه زمانی سلسله مراتبی که برگرفته از علوم زیستی است، می‌تواند جایگزین روش‌های شناسایی زبان اشاره جدید باشد زیرا توانسته است کارایی خوبی را از خود نشان دهد. نه فقط برای شناسایی زبان اشاره، بلکه برای سایر کاربردها هم می‌توان از آن بهره برد[4,10] .

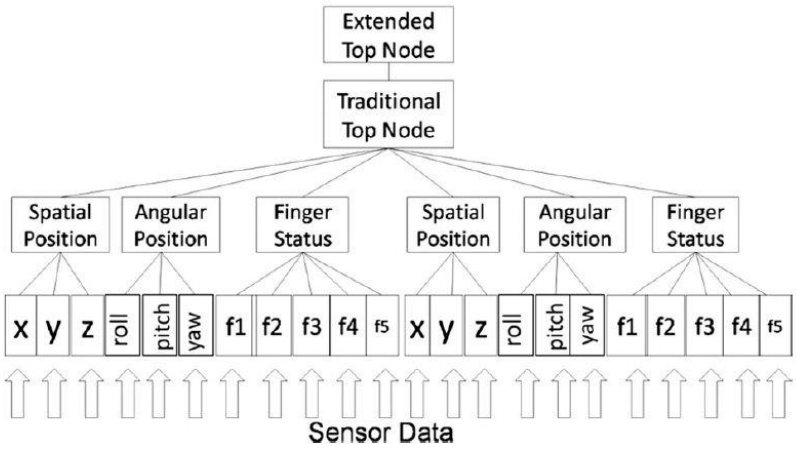
۲.۶. اصلاح حافظه

حافظه زمانی سلسله مراتبی استاندارد که معرفی کردیم دارای مشکلاتی بوده است که باعث می‌شود عملکرد آن خوب نباشد. این مشکل به این صورت است که پیش‌بینی‌هایی که شبکه انجام می‌دهد در هر لحظه زمانی انجام می‌شود و این یک عیب محسوب می‌شود؛ زیرا الگوهایی که وارد می‌شوند ممکن است در یک لحظه که پیشی بینی انجام می‌شود، هنوز کامل نشده باشند. پس استنتاج تا زمانی که بازنمایی تقریباً کاملی نداشته باشیم، بی‌معنی خواهد بود. به همین دلیل در این پیاده‌سازی، به اصلاح ساختار این شبکه پرداخته‌اند و با اضافه کردن یک لایه سطح بالاتر از آخرین لایه حافظه، عملکرد این شبکه را بهبود داده‌اند. شکل ۱۶ ساختار اصلی حافظه زمانی سلسله مراتبی را نشان می‌دهد [1,3,9]؛ و شکل ۱۷، ساختار جدید حافظه زمانی سلسله مراتبی توسعه‌یافته‌ای را نمایش می‌دهد که در بالای لایه آخر مدل استاندارد یک لایه دیگر اضافه کرده‌اند.[1]



شکل ۱۶ – نمایش ساختار سلسله مراتبی برای حافظه زمانی سلسله مراتبی استاندارد [1,3,9]

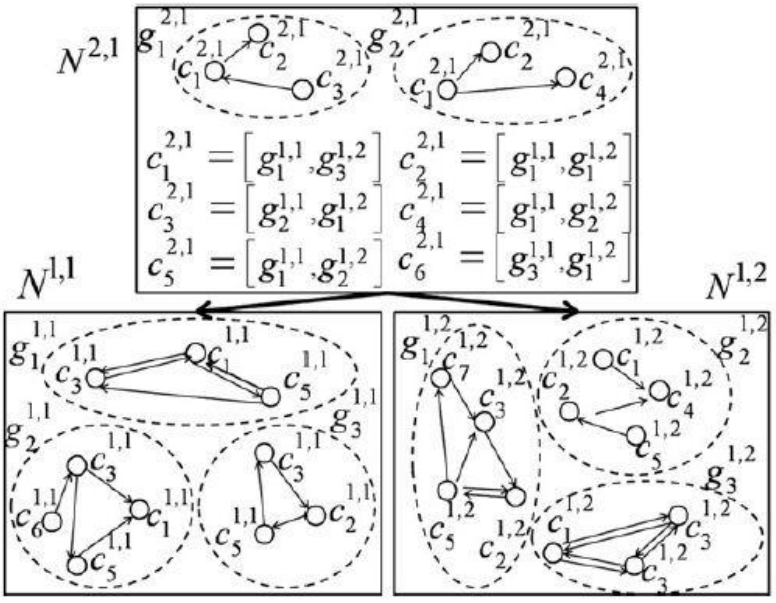
این لایه سطح بالا وظیفه دارد که پیش‌بینی‌ها را در بازه‌های زمانی خاصی انجام دهد و نه در هر لحظه. این کار باعث می‌شود که هم تعداد پیش‌بینی‌های آخرین لایه کمتر شود و همچنین بعد از اینکه الگو به صورت تقریباً کامل وارد شبکه شد، پیش‌بینی انجام شود.



شکل ۱۷ – مدل توسعه‌یافته حافظه زمانی سلسله مراتبی برای کاربرد شناسایی زبان اشاره [1]

معمولاً در روش‌هایی که قبلاً استفاده می‌شده است، یا حرکات را دسته‌بندی می‌کرده‌اند و یا اینکه از یادگیری با نظارت در آموزش شبکه استفاده شده ولی در این پیاده‌سازی، با اضافه کردن این لایه سطح بالا، مواردی که گفته شد را (یادگیری با نظارت و دسته‌بندی اولیه حرکات) در این شبکه به صورت ضمنی پیاده کرده‌اند [1].

در شکل ۱۸ نشان داده می‌شود که در این پیاده‌سازی، سعی می‌شود که بر اساس اینکه سلول‌ها چند بار پشت سرهم فعال می‌شوند (اینکه ورودی‌هایی که می‌آیند چقدر به هم مرتبط باشند) این سلول‌ها نیز در یک ناحیه فعال خواهند شد و به هم اتصالاتی را دارند. این روال دقیقاً همان پیش‌بینی آینده از روی الگوهای قبلی را به ما می‌گوید.

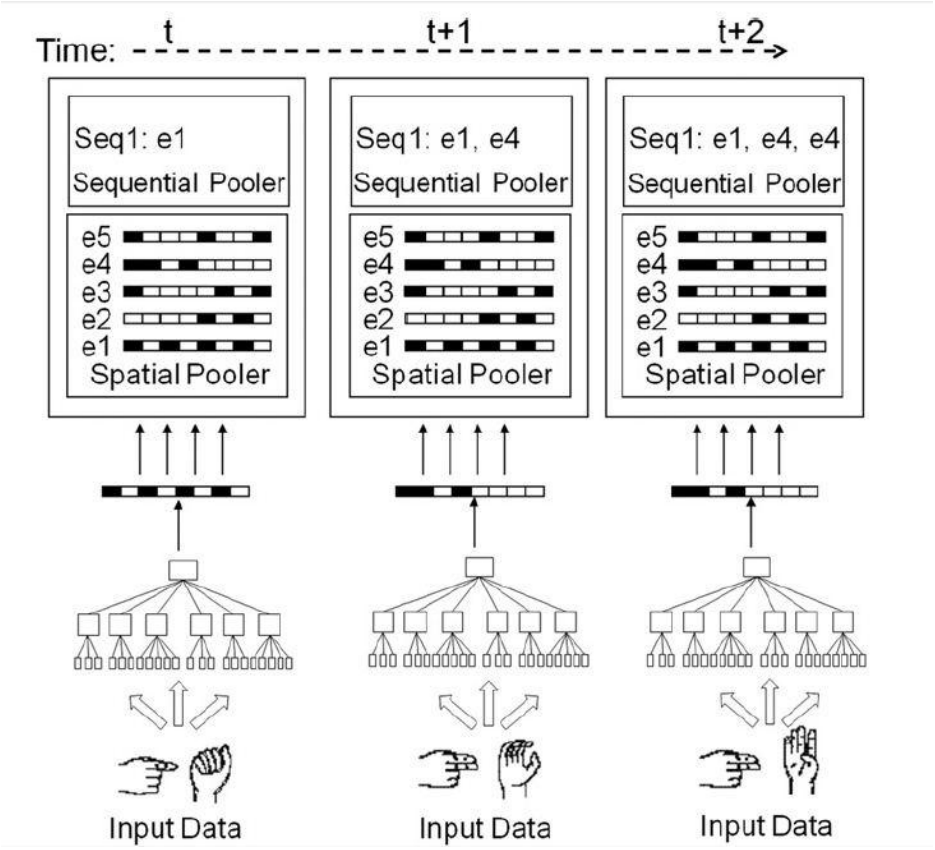


شکل ۱۸ – مدل حافظه دولایه‌ای و نحوه ارتباطات پیش‌خور و عرضی سلول‌های آن[1]

پس تنها تفاوت این ساختار با ساختار استاندارد در نود آخری است که به این شبکه اضافه شده است. این نود یک یادگیری با نظارت را شبیه‌سازی می‌کند. لایه آخر حافظه استاندارد همچون لایه‌های دیگر، تجمیع داده‌ها ندارد و فقط ورودی که از لایه فرزند به آن ارسال می‌شود را به دسته‌ای که مشابه است نگاشت می‌کند. این نگاشت در هر لحظه از زمان اتفاق می‌افتد درصورتی‌که لایه اضافه شده در بالای آن، همین نگاشت را در بازه‌های زمانی خاصی انجام می‌دهد[1] . در این حافظه توسعه‌یافته از مدل مخفی مارکوف هم استفاده شده است. مدل مخفی مارکوف یکی از روش‌هایی است که برای شناسایی زبان اشاره و بازشناسی گفتار نیز کاربرد دارد که در بازشناسی گفتار، کارایی

بسیار بالایی برای آن دیده شده است[8] . مارکوف یک مدل مولد احتمالاتی است که باعث می‌شود حافظه ما یک مدل مولدی باشد. مدل مولدی چنین کار می‌کند که به هر دنباله یک برچسب میزند و این باعث کمک به عملکرد حافظه زمانی سلسله‌مراتب می‌شود [12].

و همچنین برای فیدبک لایه‌ها از مدل پس انتشار اعتقادی بیزین استفاده شده است[1]. در آخرین لایه اضافه شده از مدل تی-کلاس برای آموزش بهره گرفته‌اند، به این صورت که زمانی که یک الگو وارد شبکه می‌شود، این الگو به سمت لایه‌های بالاتر می‌رود تا نهایتاً به این لایه پایانی برسد. آنگاه هر رویداد (مجموعی از رویدادهایی که در لایه آخر حافظه استاندارد) که به این ناحیه می‌رسد، این ناحیه به زیررویدادهایی که ذخیره کرده است و از قبل مشخص شده است نگاه می‌کند. سپس بعد از آنالیز این زیر رویدادها از میان آن‌ها بهترین و مهم‌ترین‌ها را انتخاب می‌کند؛ یعنی با آمدن رویداد جدید، این لایه به زیررویدادهای قبلی نگاه می‌اندازد و پیش‌بینی این ناحیه بر اساس رویدادهای قبلی خواهد بود. به‌طورکلی می‌توان گفت این آموزش بر اساس بودن یا نبودن رویداد جدید در زیررویدادهای قبلی است. شکل ۱۹ نشان می‌دهد که لایه آخر در هر لحظه پیش‌بینی را انجام نمی‌دهد بلکه در یک بازه زمانی خاص، پیش‌بینی خود را ارائه می‌دهد.

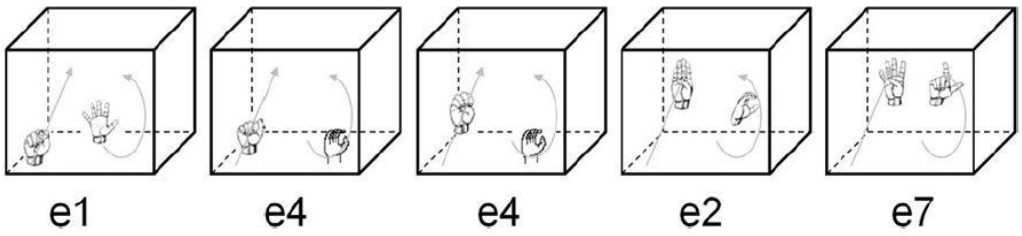


شکل ۱۹ - نحوه پیش‌بینی در لایه آخر اضافه شده در حافظه زمانی سلسله مراتبی[1]

درنتیجه می‌توان گفت که نود بالای این شبکه که در شکل ۱۸ و ۱۹ نشان داده شده است، سعی دارد که وابستگی محلی زیادی که بین سلول‌های حافظه زمانی سلسله مراتبی هست را رفع کند؛ یعنی همان پیش بنی در بازه‌های زمانی خاص و نه در هر لحظه زمانی. به صورت کلی می‌توان گفت وظیفه این نود، نوعی انتزاع است به‌گونه‌ای که دنباله‌های زمانی مناسب را در خود نگه دارد که این دنباله‌ها یک شروع و یک پایان منطقی را داشته

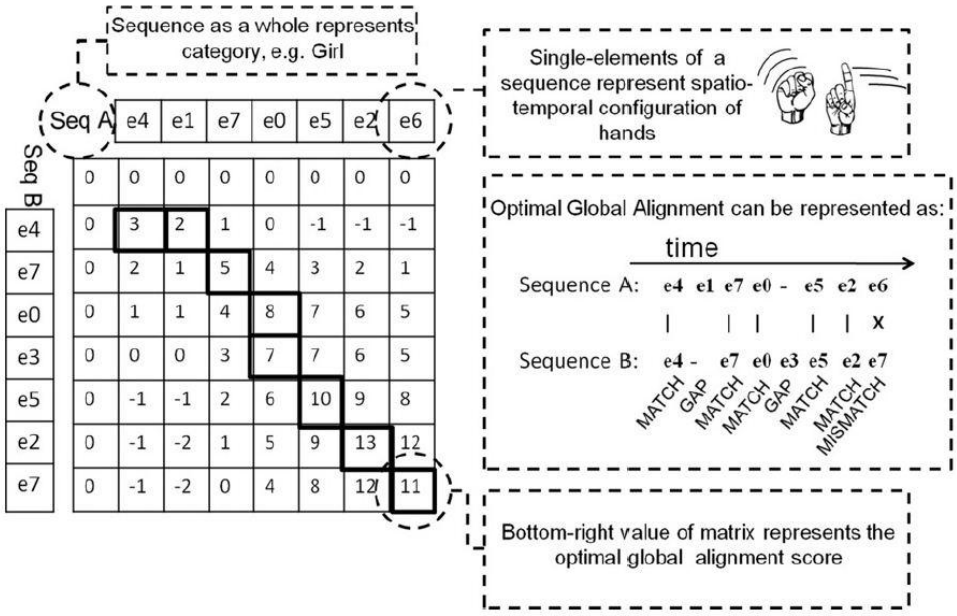
[1] Bayesian belief propagation

باشند. به همین دلیل در پیاده‌سازی این مقاله، علاوه بر متمرکز کننده زمانی و مکانی که جز اصول حافظه بود، یک متمرکز کننده دنباله‌ای را نیز در لایه آخر قرار داده‌اند.



شکل ۲۰ – یک رویداد با شروع و پایان[1]

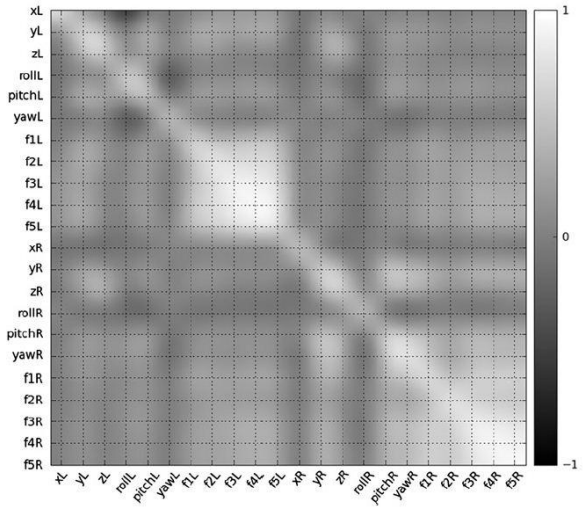
این متمرکز کننده سعی می‌کند که یک دنباله ورودی که از لایه آخر حافظه استاندارد می‌آید را با الگوهای ذخیره‌شده خود مطابقت دهد. چون این دنباله ممکن است در طول زمان متغیر باشد به همین دلیل برای این مطابقت از روش نیدلمن استفاده شده است. با استفاده از این روش می‌توان مشابه‌ترین دنباله ورودی را با دنباله‌های ذخیره‌شده پیدا کرد و به عنوان پیش‌بینی این ناحیه در نظر گرفت. از طرفی به علت اینکه تعداد دنباله‌های ذخیره‌شده برای این ناحیه می‌تواند بسیار زیاد از حد باشد، یک راه مناسب برای این مشکل گروه‌بندی دنباله‌های مشابه است. با این کار، جستجوی الگوهای ورودی ساده‌تر و حافظه مورد استفاده کمتر خواهد شد. شکل ۲۱ به صورت کلی، نمایش استفاده از الگوریتم نیدلمن را در لایه آخر نشان می‌دهد [1].



شکل ۲۱ – نحوه استفاده از روش نیدلمن برای آخرین لایه اضافه شده در شبکه [1]

همان‌گونه که گفته شد، آخرین نودی که اضافه شده بود، برای حل وابستگی زیاد محلی بین لایه‌ها و سلول‌های حافظه زمانی سلسله مراتبی است؛ زیرا شبکه سعی دارد که بین سلول‌ها خود وابستگی ایجاد کند که هم یادگیری را داشته باشد و هم پیش‌بینی و این وابستگی ضروری است. هرچند در این پیاده‌سازی با اضافه کردن یک نود در آخرین سطح، این وابستگی که در هر لحظه از زمان رخ می‌دهد را به بازه‌های زمانی خاص تبدیل کردیم تا از پیش‌بینی‌هایی که در هر لحظه زمانی انجام می‌شود دوری کنیم. شکل ۲۲، کانال‌های ورودی را به صورت ماتریس دوبعدی نشان داده‌اند. در این ماتریس مشخص است که بعضی از نواحی در ارتباط با هم دیگر هستند. این ارتباط‌ها

به صورت محلی خود را نشان داده‌اند. به عنوان مثال کانال‌های ورودی برای انگشتان، یک محدوده خاص را که نشان از وابستگی محلی بین آن‌ها دارد را مشخص کرده است.

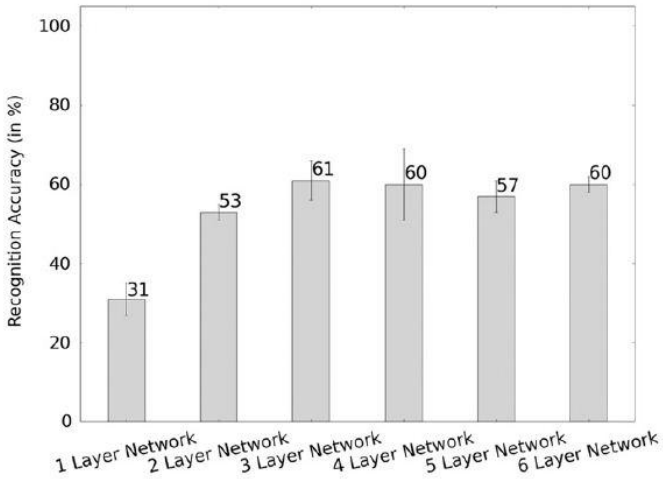


شکل ۲۲ – ماتریس وابستگی محلی بین حسگرهای ورودی و سلول‌های شبکه [1]

به صورت خلاصه این‌چنین می‌توان گفت که برای بهبود کار شبکه و دوری کردن از وابستگی‌های محلی بین لایه‌ها و سلول‌ها، یک لایه بالای لایه اصلی شبکه استاندارد اضافه شده است. این لایه به نوعی یادگیری با نظارت را در بازه‌های زمانی خاصی انجام می‌دهد جایی که اول و پایان بازه‌ها مشخص است. از مدل مخفی مارکوف و همچنین قوانین بازخورد برای بالا بردن کارایی استفاده شده است. در بخش بعدی نتایجی که به دست آمده است را مرور می‌کنیم تا مقایسه‌ای بین این شبکه و سایر شبکه‌های موجود برای چنین مسائل خاص را داشته باشیم[1] .

۳.۶. نتایج

نتایجی که در این مقاله از این پیاده‌سازی به دست آمده است را به صورت خلاصه مطرح می‌کنیم. برای اولین نتیجه، شکل نمودار ۲۳ را در نظر بگیرید. در این شکل متوجه می‌شویم که تعداد لایه‌های این شبکه، هر چه بالاتر برود دلیل بر بهتر شدن کارکرد شبکه نخواهد بود و در این کاربرد شبکه با سه لایه کافی بوده است. البته همان‌طور که در اوایل گزارش گفتیم، شبکه مغزی انسان ۶ لایه بوده است و حافظه پیاده‌سازی شده در این مقاله هم برای سه لایه و هم برای ۶ لایه عملکرد تقریباً مشابهی داشته‌اند[1,2] .



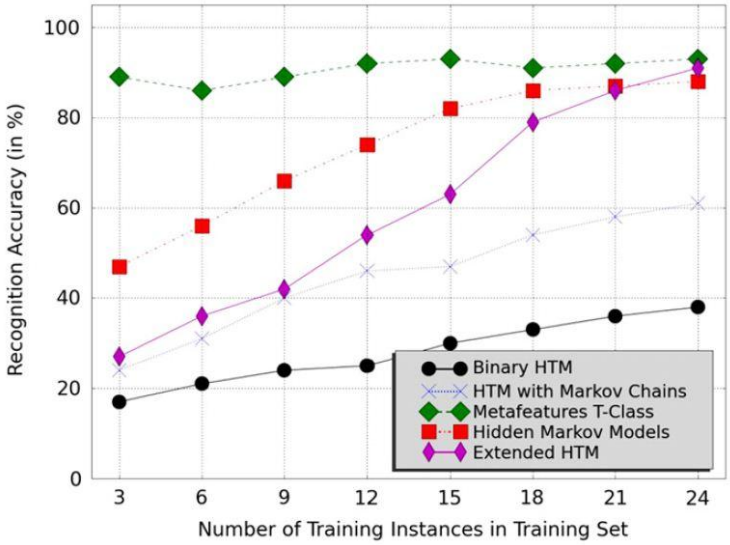
شکل ۲۳ – عملکرد شبکه برای توپولوژی‌های متفاوت لایه‌های شبکه[1]

شکل ۲۴، مقدار صحت عملکرد انواع حافظه‌های پیاده‌سازی شده و نیز مدل‌های دیگر روش حل این کاربرد را تخمین نشان می‌دهد. همان‌گونه که مشاهده می‌شود، بهترین عملکرد مربوط به تی-کلاس است با ۹۳ درصد صحت. مدل مخفی مارکوف که یک روش متداول در این کاربرد است دقت ۸۸ درصدی دارد ولی حافظه زمانی سلسله مراتبی که با مکانیزم‌ها و روش‌هایی ترکیب شده است، در بهترین حالت توانسته صحت ۹۱ درصد را بدهد که بهتر از مدل مخفی مارکوف بوده است.

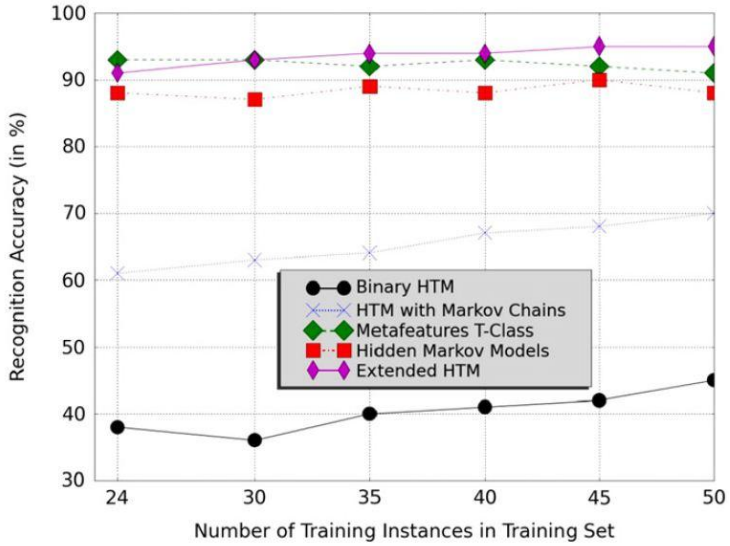
Row	Simulation type	Accuracy
1	Binary HTM (Gen)	23%
2	Binary HTM (Spe)	27%
3	Binary HTM (Best)	38%
4	Markovian HTM	61%
5	Metafeatures TClass	93%
6	Hidden Markov Models	88%
7	Extended HTM with DP (Gen)	73%
8	Extended HTM with DP (Spe)	74%
9	Extended HTM with DP (Rha)	70%
10	Extended HTM with DP (Der)	57%
11	Extended HTM with DP (De2)	38%
12	Extended HTM with DP (Int)	59%
13	Extended HTM with DP (Ave)	56%
14	Combined Extended HTM (Gen+Spe)	82%
15	Combined Extended HTMs (Gen+Spe+Rha)	85%
16	Combined Extended HTMs (Gen+Spe+Rha+Der+De2+Int+Ave)	91%

شکل ۲۴ – صحت عملکرد شناسایی زبان اشاره برای روش‌های مختلف

شکل ۲۵ و ۲۶ چنین میگویند که حافظه زمانی سلسله مراتبی نیاز به آموزش زیاد دارد [1]. در این شکل‌ها به وضوح دیده می‌شود که اگر تعداد نمونه‌های آموزشی کم باشند حافظه زمانی نسبت به مدل مارکوف و یا تی-کلاس دارای عملکرد خوبی نیست ولی به‌مرورزمان و با بالا رفتن تعداد نمونه‌های آموزشی شبکه به سمت دقت بالاتر می‌رود و حتی جایی می‌رسد که از عملکرد مارکوف و تی-کلاس نیز برتر می‌شود.



شکل ۲۵ – درصد صحت عملکرد شبکه تعداد داده‌های آموزشی در هر مجموعه آموزشی – بخش اول [1]



شکل ۲۶- درصد صحت عملکرد شبکه تعداد داده‌های آموزشی در هر مجموعه آموزشی – بخش دوم [۱]

۷. نتیجه

همان‌گونه که گفته شد، حافظه زمانی سلسله مراتبی قصد دارد که شیوه کار مغز انسان را به ساده‌ترین شکل ممکن و با بالاترین دقت عملکرد، شبیه‌سازی کند. هرچند پیاده‌سازی‌هایی انجام شده است، ولی بازهم پیچیدگی‌های این شبکه پابرجاست. با این وجود، دقت این شبکه در کاربردهایی که با زمان متغیر هستند، می‌تواند بالاتر از روش‌های دیگر باشد.

البته بر اساس مثالی که از کاربرد این شبکه زدیم، نسخه استاندارد حافظه زمانی سلسله مراتبی دارای نواقصی است که خود نویسندگان و کارمندان شرکت نومنتا نیز آن را اعلام کرده بودند که الگوریتم‌های پایه این شبکه جای پیشرفت و توسعه دارد و در صدد پیشرفت این شبکه هستند. همان‌طور که برای کاربرد مثال زده شده، یک نود پایانی دیگر به شبکه اضافه شد تا مشکل پیش‌بینی در هر لحظه زمانی حل شود و یا با اضافه کردن خصوصیات و روش‌هایی (مدل مخفی مارکوف، آموزش تی-کلاس و الگوریتم نیدلمن) به حافظه، سعی در بهبود عملکرد شبکه را داشته‌اند؛ و در نتایجی که از کاربرد شناسایی زبان اشاره آوردیم مشاهده کردیم که این شبکه می‌تواند جایگزین روش‌های قبلی در این کاربرد باشد و نه حتی در این کاربرد بلکه در کاربردهای دیگر نیز، همچون شناسایی صوت و یا تخمین سری‌های زمانی و در کاربردهایی که با ویدیو در ارتباط است می‌تواند مفید باشد.

توسعه‌هایی که می‌توان برای این حافظه انجام داد، بحث اصلاح و یا انتخاب قوانین مناسب بازخورد و یا نحوه اتصال و دسته‌بندی سلول‌های داخل یک ناحیه و درصورتی‌که بتوانیم با اضافه کردن یک لایه سطح پایین در حافظه که بتواند پیش‌پردازش‌های خاصی را روی داده‌های ورودی انجام دهد باعث بالا رفتن کارایی این شبکه خواهد شد، همان‌گونه که در روش‌های شناسایی زبان اشاره که پیش از این پیاده شده بودند، با انجام پیش‌پردازش روی داده‌های ورودی و دسته‌بندی اولیه آن‌ها، کارایی و دقت روش‌های بالاتر می‌رفت تا مثل مقاله حاضر بدون هیچ‌گونه پیش‌پردازشی روی داده‌های ورودی، مستقیماً داده‌ها را در اختیار حافظه زمانی سلسله مراتبی قرار می‌دهیم.

مراجع

[1] D.Rozado, F.B.Rodriguez,P.Varona, "Extending the bioinspired hierarchical temporal memory paradigm for sign language recognition", Neurocomputing, Vol 79,PP 75–86, 2012.

[2] Numenta Inc, "Hierarchical Temporal Memory Including HTM Cortical Learning Algorithms", VERSION 0.2.1, 2011.

[3] Numenta Inc. J.Hawkins,D.George, "Hierarchical Temporal Memory, Concepts, Theory, and Terminology", 2006.

[4] D.Rozado, F.B.Rodriguez, P.Varona, "Gaze Gesture Recognition with Hierarchical Temporal Memory Networks", Advances in Computational Intelligence, Vol 6691, PP1-8, 2011.

[5] D.Maltoni, E.M.Rehn, "Incremental Learning by Message Passing in Hierarchical Temporal Memory", Journal Neural Computation, Vol 26, Issue 8, PP 1763-1809, 2014, MIT Press Cambridge, MA, USA

[6] S.B.Needleman, C.D.Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", Journal of Molecular Biology, Vol 48, Issue 3, PP 443–453, 1970

[7] A.Karami, B.Zanj, A.K.Sarkaleh, "Persian Sign Language (PSL) Recognition Using Wavelet Transform and Neural Networks", Expert Systems with Applications, Vol 38, PP 2661–2667, 2011.

[8] V.V.Digalakis, O.A. Kimball, "From HMM's to segment models: a unified view of stochastic modeling for speech recognition Speech and Audio Processing", IEEE Transactions on, Vol 4, Issue 5, PP 360–378, 1996.

[9] R.Skoviera, I.Bajla, "Image Classification Based on Hierarchical Temporal Memory and Color Features", Institute of Measurement Science.

[10] D.Maltoni, "Pattern Recognition by Hierarchical Temporal Memory", DEIS Technical Report, Università degli Studi di Bologna, Biometric System Laboratory, 2011.

[11] R.J.Rodriguez, J.A.Cannady, "Towards a hierarchical temporal memory based self-managed dynamic trust replication mechanism in cognitive mobile ad-hoc networks", AIKED'11 Proceedings of the 10th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases, Stevens Point, Wisconsin, USA, PP 320-328, 2011

[12] "Generative model", Wikipedia, Retrieved January, 2014, from: http://en.wikipedia.org/wiki/Generative_model

Supporting data structures and routines for Partial Pooler

input(t,j)
The input to this level at time t. input(t, j) is 1 if the j'th input is on.

overlap(*c*)
The spatial pooler overlap of column c with a particular input pattern.

activeColumns(*t*)
List of column indices that are winners due to bottom-up input.

desiredLocalActivity
A parameter controlling the number of columns that will be winners after the inhibition step.

desiredLocalActivity
A parameter controlling the number of columns that will be winners after the inhibition step

inhibitionRadius
Average connected receptive field size of the columns.

neighbors(c)
A list of all the columns that are within inhibitionRadius of column c.

minOverlap
A minimum number of inputs that must be active for a column to be considered during the inhibition step.

boost(c)
The boost value for column c as computed during learning - used to increase the overlap value for inactive columns.

synapse
A data structure representing a synapse - contains a permanence value and the source input index.

connectedPerm
If the permanence value for a synapse is greater than this value, it is said to be connected.

potentialSynapses(c)
The list of potential synapses and their permanence values.

connectedSynapses(c)
A subset of potentialSynapses(c) where the permanence value is greater than connectedPerm. These are the bottom-up inputs that are currently connected to column c.

permanenceInc
Amount permanence values of synapses are incremented during learning.

permanenceDec
Amount permanence values of synapses are decremented during learning.

activeDutyCycle(c)
A sliding average representing how often column c has been active after inhibition (e.g. over the last 1000 iterations)

overlapDutyCycle(c)
A sliding average representing how often column c has had significant overlap (i.e. greater than minOverlap) with its inputs (e.g. over the last 1000 iterations).

minDutyCycle(c)
A variable representing the minimum desired firing rate for a cell. If a cell's firing rate falls below this value, it will be boosted. This value is calculated as 1% of the maximum firing rate of its neighbors.

kthScore(cols, k)
Given the list of columns, return the k'th highest overlap value.

updateActiveDutyCycle(c)
Computes a moving average of how often column c has been active after inhibition.

updateOverlapDutyCycle(c)
Computes a moving average of how often column c has overlap greater than minOverlap.

averageReceptiveFieldSize()
The radius of the average connected receptive field size of all the columns. The connected receptive field size of a column includes only the connected synapses (those with permanence values >= connectedPerm). This is used to determine the extent of lateral inhibition between columns.

maxDutyCycle(cols)
Returns the maximum active duty cycle of the columns in the given list of columns.

increasePermanences(c, s)
Increase the permanence value of every synapse in column c by a scale factor s.

boostFunction(c)
Returns the boost value of a column. The boost value is a scalar >= 1. If activeDutyCyle(c) is above minDutyCycle(c), the boost value is 1. The boost increases linearly once the column's activeDutyCyle starts falling below its minDutyCycle

The following data structures are used in the temporal pooler pseudocode:

cell(c,i)
A list of all cells, indexed by *i* and *c*.

cellsPerColumn
Number of cells in each column.

activeColumns(*t*)
List of column indices that are winners due to bottom-up input (this is the output of the spatial pooler).

activeState(c, i, t)
A boolean vector with one number per cell. It represents the active state of the column c cell i at time t given the current feed-forward input and the past temporal context. activeState(c, i, t) is the contribution from column c cell i at time t. If 1, the cell has current feed-forward input as well as an appropriate temporal context.

predictiveState(c, i, t)
A boolean vector with one number per cell. It represents the prediction of the column c cell i at time t, given the bottom-up activity of other columns and the past temporal context.

predictiveState(c, i, t)
is the contribution of column c cell i at time t. If 1, the cell is predicting feed-forward input in the current temporal context.

learnState(c, i, t)
A boolean indicating whether cell i in column c is chosen as the cell to learn on.
activationThreshold Activation threshold for a segment. If the number of active connected synapses in a segment is greater than activationThreshold, the segment is said to be active.
learningRadius The area around a temporal pooler cell from which it can get lateral connections.

initialPerm Initial
permanence value for a synapse.

connectedPerm
If the permanence value for a synapse is greater than this value, it is said to be connected.

minThreshold
Minimum segment activity for learning.

newSynapseCount
The maximum number of synapses added to a segment during learning.

permanenceInc
Amount permanence values of synapses are incremented when activity-based learning occurs.

permanenceDec
Amount permanence values of synapses are decremented when activity-based learning occurs

segmentUpdate
Data structure holding three pieces of information required to update a given segment: a(segment index (-1 if it's a new segment), b) a list of existing active synapses, and c) a flag indicating whether this segment should be marked as a sequence segment (defaults to **false**).

segmentUpdateList
A list of segmentUpdate structures. segmentUpdateList(c,i) is the list of changes for cell i in column c.

segmentActive(s, t, state)
This routine returns **true** if the number of connected synapses on segment s that are active due to the given state at time t is greater than activationThreshold. The parameter state can be activeState, or learnState.

getActiveSegment(c, i, t, state)
For the given column c cell i, return a segment index such that segmentActive(s,t, state) is **true**. If multiple segments are active, sequence segments are given preference. Otherwise, segments with most activity are given preference.

getBestMatchingSegment(c, i, t)
For the given column c cell i at time t, find the segment with the largest number of active synapses. This routine is aggressive in finding the best match. The permanence value of synapses is allowed to be below connectedPerm. The number of active synapses is allowed to be below activationThreshold, but must be above minThreshold. The routine returns the segment index. If no segments are found, then an index of -1 is returned.

getBestMatchingCell(c)
For the given column, return the cell with the best matching segment (as defined above). If no cell has a matching segment, then return the cell with the fewest number of segments.
getSegmentActiveSynapses(c, i, t, s, newSynapses= **false**)
Return a segmentUpdate data structure containing a list of proposed changes to segment s. Let activeSynapses be the list of active synapses where the originating cells have their activeState output = 1 at time step t. (This list is empty if s = -1 since the segment doesn't exist.) newSynapses is an optional argument that defaults to **false**. If newSynapses is **true**, then newSynapseCount - count(activeSynapses) synapses are added to activeSynapses. These synapses are randomly chosen from the set of cells that have learnState output = 1 at time step t.

adaptSegments(segmentList, positiveReinforcement)
This function iterates through a list of segmentUpdate's and reinforces each segment. For each segmentUpdate element, the following changes are performed. If positiveReinforcement is **true** then synapses on the active list get their permanence counts incremented by permanenceInc. All other synapses get their permanence counts decremented by permanenceDec. If positiveReinforcement is **false**, then synapses on the active list get their permanence counts decremented by permanenceDec. After this step, any synapses in segmentUpdate that do yet exist get added with a permanence count of initialPerm.