مروری بر الگوریتم‌های شبکه‌های عصبی

فرزاد عبدالحسینی، سید سبحان میریوسفی، هومن هاشمی
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف
{abdolhosseini, miryoosefi, hohashemi}@ce.sharif.edu

چکیده

در این مقاله قرار است نگاهی به الگوریتم‌های شبکه‌های عصبی، تاریخچه و کارهای انجام شده در آن داشته باشیم. با مفاهیم پایه‌ای آن آشنا شده و الگوریتم‌ها و روش‌های به کار رفته را به طور اجمالی بررسی کنیم. فرض شده که مخاطب آشنایی زیادی با این رشته ندارد و بیشتر می‌خواهد یک دید اولیه از آن به دست آورد. در ابتدا مفاهیم اولیه مورد نیاز همانند تعریف مدل نورون آمده و سپس به بررسی انواع معماری‌های شبکه‌های عصبی پرداخته می‌شود. بعد از آن الگوریتم‌های مختلف یادگیری تا حدی بررسی شده و در انتها بعضی از کاربردهای مهم عملی این شبکه‌ها نشان‌داده می‌شود. به طور خلاصه، دلیل اصلی استفاده از شبکه‌های عصبی مصنوعی در عمل، سرعت و یا قدرت بالای آن‌ها در پردازش نیست. بلکه توانایی یادگیری و تطبیق‌پذیری آن‌ها با مسائل مختلف است که این شبکه‌ها را ار باقی الگوریتم‌ها جدا می‌کند.

کلید واژه‌ها_ الگوریتم، هوش مصنوعی، شبکه‌های عصبی.

۱ـ مقدمه

کار بر روی **شبکه های عصبی مصنوعی** یا به اختصار **شبکه های عصبی** از جایی آغاز شد که دانشمندان به این مهم دست یافتند که سیستم پردازش مغز انسان بسیار متفاوت با سیستم‌های کامپیوتری دیجیتال مرسوم میباشد. مغز انسان از یک ساختار بسیار پیچیده، غیر خطی و موازی بهره میبرد و همچنین قابلیت بهبود و ارتقای خود را نیز دارد.

برای مثال قدرت بینایی و بصری انسان را در نظر بگیرید که یک نوع پردازش اطلاعات است. ما اطلاعات را از محیط بیرون توسط حسگر پیچیده چشم دریافت کرده آن هارا تجزیه و تحلیل میکنیم تا بتوانیم چیز هایی که برای تعامل با محیط نیاز داریم بدست آوریم. مغز انسان میتواند فرآیند تشخیص چهره در یک محیط نا آشنا را در کمتر از ۲۰۰ میلی‌ثانیه انجام دهد در صورتی که فرآیند های بسیار ساده تر از این برای کامپیوتر های حال حاضر چند روز زمان میبرد.

به عنوان مثالی دیگر خفاش را در نظر بگیرید. این خفاش از یک سیستم ردیاب صوتی یا همان سونار بهره میبرد به این شکل که هنگامی که در تعقیب شکار خود است یک موج صوتی از خود ساطع میکند و از روی انکعاس آن توسط طعمه میتواند اطلاعاتی نظیر سرعت نسبی طعمه، اندازه طعمه و ... بدست آورد. تمام این پردازش های پیچیده در مغز کوچک خفاش که به اندازه‌ی یک آلو است انجام می‌پذیرد.

اما مغز انسان یا خفاش چگونه این کار هارا انجام می‌دهد ؟

۲ـ آشنایی و مفاهیم اولیه

در اینجا ابتدا یک تعریف ارائه میکنیم[۱، ص_۲] و سپس در بخش های بعد به توضیح آن میپردازیم:

شبکه عصبی مصنوعی یک پردازنده توزیع شده موازی و گسترده است و از واحد های پردازشی ساده ساخته شده که میتواند دانش اکتسابی را در خود ذخیره و در آینده از آن در تصمیم‌گیری‌ها استفاده کند. از دو جهت این شبکه عصبی مصنوعی مغز انسان را تداعی می‌کند:

۱. **اطلاعات از محیط و توسط فرآیند یادگیری کسب می‌شود.**

۲. **میزان قدرت اتصالات بین واحد های پردازشی(نورون ها) که به آن وزن سیناپسی میگوییم برای ذخیره اطلاعات استفاده می‌شود.**

۲ـ۱ـ مزایای شبکه‌های عصبی

شبکه عصبی قدرت محاسباتی‌اش را از دو ویژگی بهره می‌گیرد ویژگی اول اینکه یک شبکه بسیار گسترده، موازی و توزیع شده است و ویژگی دوم قابلیت یادگیری است. این که می‌تواند با توجه به محیط یک سری اطلاعات و تجارب کسب کند و از این تجارب در تصمیم‌گیری‌های بعدی استفاده کند به این ویژگی **کلیت‌بخشی**[۱] می‌گوییم. این ویژگی ها باعث می‌شود مسائلی که در حال حاضر برای کامپیوترهای مرسوم دست‌نیافتی است توسط این شبکه‌ها

[۱] Generalization

قابل حل باشد البته باید توجه داشت که گونه‌های خاصی از مسائل هستند که توسط این شبکه‌های عصبی به طور بهینه قابل حل نیستند و پردازنده‌های مرسوم در بعضی موارد بهتر عمل می‌کنند. همچنین باید توجه داشت که باید فقط بخش‌هایی از یک مسئله که مرتبط با شبکه‌های عصبی هستند را توسط این شبکه‌ها حل کنیم و برای باقی از روش‌های مناسب دیگر بهره بجوییم. در نهایت لازم به ذکر است که تا ساختن شبکه عصبی‌ای که به طور کامل مغز انسان یا هر شبکه عصبی طبیعی دیگر را تداعی کند راه طولانی‌ای در پیش داریم.

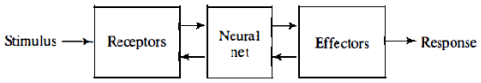
حالا بعضی از مزایا و توانایی‌های شبکه‌های عصبی را ذکر می‌کنیم[۱، ص–۳]:

- **توانایی غیرخطی بودن**[۲]: واحدهای سازنده شبکه عصبی یا همان نورون‌ها می‌توانند غیرخطی باشند و درنتیجه کل شبکه عصبی غیرخطی می‌شود این ویژگی بسیار مهم است زیرا بسیاری از مسائل ذاتا غیرخطی هستند مانند پردازش گفتار.

- **نگاشت ورودی–خروجی**[۳]: در بسیاری از مسائل ما ارتباط منطقی بین ورودی و خروجی را نمیدانیم ولی برای چند نمونه خاص ورودی خروجی مورد انتظار را در دست داریم ما می‌توانیم به کمک این‌ها شبکه عصبی را آموزش دهیم در اینصورت شبکه عصبی وزن‌های سیناپسی خود را به گونه‌ای تغییر می‌دهد که جواب تا حد ممکن به خروجی مورد نظر نزدیک شود در اینصورت به گونه‌ای توانستیم بدون هیچ دانسته قبلی، بین ورودی و خروجی‌های درست نگاشت برقرار کنیم حال می‌توانیم از شبکه عصبی برای بدست آوردن خروجی مورد نظر برای ورودی‌های دیگر استفاده کنیم این تکنیک در مسائل الگویابی بسیار استفاده می‌شود.

- **تطبیق‌پذیری**[۴]: ساختار شبکه عصبی به نحوی است که می‌تواند خود را در محیط وفق دهد به اینگونه که اگر محیط تغییر کند شبکه عصبی وزن‌های سیناپسی خود را به گونه این تغییر می‌دهد که با محیط تغییر یافته تطبیق پیدا کند البته باید توجه داشت که شبکه عصبی نباید بیش از حد هم نسبت به تغییرات حساس باشد زیرا که تغییرات خیلی کوچک که حتی می‌تواند ناشی از خطای حسگرها باشد بر روی آن تاثیر می‌گذارد این تطبیق‌پذیری باید به گونه‌ای باشد که نه خیلی حساس باشد که پایداری سیستم به هم بریزد نه خیلی بی‌تفاوت که روند یادگیری و بهبود را مختل کند.

[۲] Nonlinearity
[۳] Input-Output Mapping
[۴] Adaptivity



شکل ۱: ساختار سه بخشی شبکه عصبی انسان (از هیکین[۱])

- **تحمل خطا**[۵]: ساختار شبکه های عصبی به علت گستردگی به گونه ایست که تحمل خطا و خرابی بالایی دارد. برای مثال با از کار افتادن یک نورون تغییر محسوسی در نتیجه حاصل نمی‌شود و آسیب باید خیلی زیاد باشد تا کار شبکه را مختل کند.

- **آزمایشات زیستی**[۶]: از آن جایی که شبکه عصبی مصنوعی تا حد خوبی می‌تواند شبکه‌های عصبی واقعی را تداعی کند از آن می‌توانیم در آزمایشات و پژوهش‌های زیستی استفاده کنیم.

۲–۲– بررسی مغز انسان

از آن جایی که شبکه عصبی برگرفته از مغز انسان است ابتدا به بررسی مغز می‌پردازیم . شبکه عصبی مغز انسان یک سیستم سه بخشی است همانطور که در شکل ۱ نشان داده شده است.

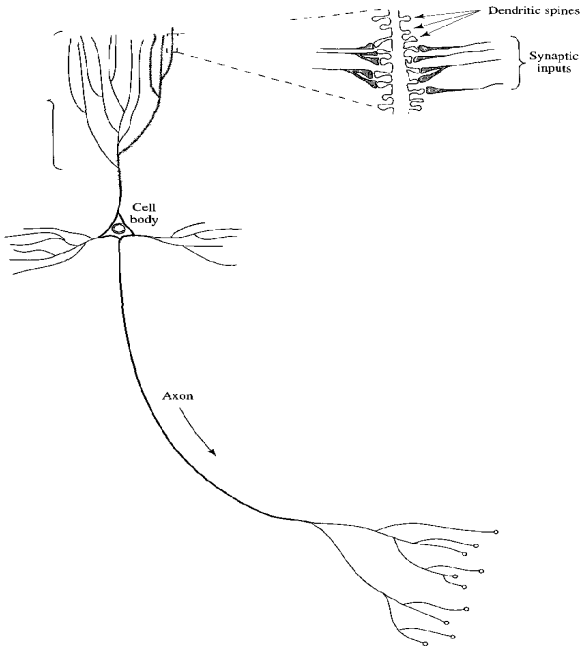
بخش مرکزی همان شبکه عصبی است که شامل شبکه گسترده نورون‌ها بوده و محل تصمیم‌گیری است. بخش اول یا همان گیرنده‌ها اطلاعات را از محیط گرفته و به سیگنال‌های قابل فهم برای شبکه عصبی تبدیل می‌کند و بخش آخر نیز دستورات را از شبکه عصبی گرفته و واکنش موردنظر را در محیط انجام می‌دهد.

در سال ۱۹۱۱ ساختار نورونی[۷] برای مغز معرفی شد. نورون‌ها ۵ الی ۶ مرتبه توانی از سیلیکون کند تر هستند. اتفاقات در چیپ‌های سیلیکونی هر ۱ نانوثانیه اتفاق میافتد و این در حالی است که در نورون هر ۱ میلی‌ثانیه می‌تواند یک پالس را از خود گذر بدهد (یا ندهد). ولی مغز انسان این سرعت کم نورون ها را با تعداد بسیار زیاد آن‌ها و اتصالات (سیناپس ها) بسیار زیادتر جبران کرده است. تخمین زده میشود که تعداد نورون‌های مغز حدود ۱۰ میلیارد و تعداد اتصالات بین آن ها حدود ۶۰ تریلیارد است. از طرفی دیگر مصرف انرژی مغز انسان در مقایسه با چیپ های سیلیکونی حدود ۱۰ مرتبه توانی کمتر است. در شکل ۲ ساختار نورون و سیناپس را مشاهده می‌کنید.

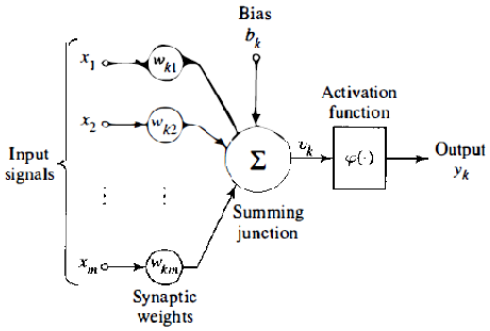
۲–۳– مدل نورون برای شبکه عصبی

نورون یک واحد پردازش اطلاعات است که واحد سازنده شبکه عصبی است در شکل ۳ مدل ارائه شده برای نورون را مشاهده

[۵] Fault Tolerance
[۶] Neurobiological Analogy
[۷] Ramon y Cajal



شکل ۲: ساختار نورون زیستی [۱]



شکل ۳: مدل ریاضی ارائه شده برای نورون [۱]

می‌کنید . در اینجا به تشریح بخش‌های مختلف می‌پردازیم:

- **سیناپس ها یا اتصالات** هر کدام به همراه یک عدد که به آن وزن سیناپسی می‌گوییم مشخص شده‌اند هنگامی که سیگنال x_i را در سیناپس j از نورون k داشته باشیم و وزن سیناپسی این سیناپس $W_{k,j}$ باشد آن وقت سیگنال ورودی در وزن سیناپسی ضرب می‌شود.

- یک **جمع کننده** که مقادیر ورودی (سیگنال های ضرب شده در وزن سیناپسی) و همچنین مقدار ثابت (bias) را جمع می‌کند.

- **تابع فعال سازی** که مقدار حاصل جمع را می‌گیرد و خروجی مورد نظر را تولید می‌کند.

همانطور که در بالا توضیح دادیم داریم:

$$u_k = \sum_{j=۱}^{k} W_{k,j} X_j$$

$$v_k = u_k + b_k$$

$$y_k = \varphi(v_k)$$

به مقدار v_k در عبارت بالا **پتانسیل فعالسازی** نورون می‌گوییم.

۲ـ۴ـ انواع توابع فعال‌سازی

- **تابع Threshold**: این تابع به نوعی بر اساس اینکه مقدار ورودی میزان مشخصی را رد کرده است یا خیر تصمیم‌گیری انجام می‌دهد و بسیار پرکاربرد است:

$$\varphi(v) = \begin{cases} ۱ & \text{if} \quad v \geq ۰ \\ ۰ & \text{if} \quad v < ۰ \end{cases}$$

- **تابع PicewiseLinear**: این تابع حالت بسیار تیز تغییر تصمیم در مدل قبلی را به کمک تابعی خطی بهبود بخشیده است:

$$\varphi(v) = \begin{cases} ۱ & \text{if} \quad v > \frac{۱}{۲} \\ v + \frac{۱}{۲} & \text{if} \quad -\frac{۱}{۲} \leq v \leq \frac{۱}{۲} \\ ۰ & \text{if} \quad v < -\frac{۱}{۲} \end{cases}$$

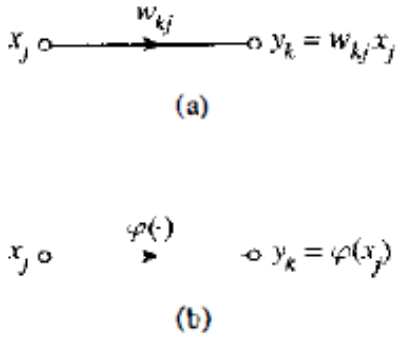
- **تابع Sigmoid**: این تابع یکی از پرکاربردترین توابع فعال سازی است در واقع چیزی بین دو حالت قبلی است. توجه کنید اگر ثابت a را به سمت بی‌نهایت میل بدهیم تبدیل به تابع Treshold می‌شود.

$$\varphi(v) = \frac{۱}{۱ + \exp(-av)}$$

- **مدل احتمالاتی**: در بعضی از مسائل نیاز است که تابع فعال سازی ما تصمیم قطعی نگیرد و به صورت احتمالاتی عمل کند مانند مثال زیر:

$$P(v) = \frac{۱}{۱ + \exp(-av)}$$

$$\varphi(v) = \begin{cases} +۱ & \text{with probability} \quad P(v) \\ -۱ & \text{with probability} \quad ۱ - P(v) \end{cases}$$



شکل ۴: قاعده ۱ [۱]

۲ـ۵ـ نمایش شبکه عصبی با گراف جهتدار

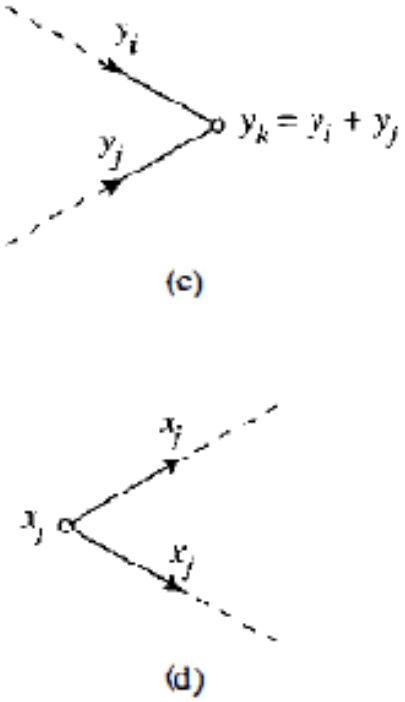
شبکه عصبی را میتوان به وسیله یک گراف جهتدار نشان داد . این گراف از سه قاعده زیر طبعیت می‌کند:

- **قاعده ۱:** سیگنال در جهت یال منتقل می‌شود. ۲ نوع یال در گراف موجود است:
  - **یال سیناپسی:** در این یال مقدار ورودی یال (سیگنال گره‌ی ابتدای یال) در وزن نوشته شده روی یال که در واقع همان وزن سیناپسی است ضرب می‌شود تا حاصل تولید شود.
  - **یال فعال‌سازی:** در این یال مقدار ورودی یال (سیگنال گره‌ی ابتدای یال) به عنوان ورودی تابع نوشته شده روی یال در نظر گرفته می‌شود تا حاصل تولید شود.
- **قاعده ۲:** سیگنال یک گره جمع جبری سیگنال یال‌هایی است که به آن وارد می‌شوند.
- **قاعده ۳:** یال‌هایی که از یک گره خارج می‌شوند از هم مستقل بوده و سیگنال اولیه برابر سیگنال گره دارند.

در شکل ۴ و ۵ می‌توانید این سه قاعده را مشاهده کنید.
در نتیجه می‌توانیم تعریفی دیگر[۱، ص‌ـ۱۷] برای شبکه عصبی ارائه دهیم:

شبکه عصبی یک گراف جهتدار تشکیل شده از گره، یال‌های سیناپسی و تابع فعال‌سازی است و دارای چهار ویژگی زیر است:

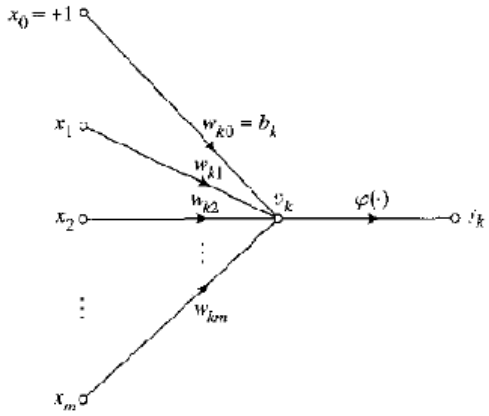
- **هر نورون متشکل از تعدادی یال سیناپسی و حداکثر یک یال فعال‌سازی است.**



شکل ۵: قاعده ۲ و ۳ [۱]

- **وزن یال‌های سیناپسی نشان دهنده وزن‌های سیناپسی است (مقادیر قابل تغییر در یادگیری).**
- **جمع سیگنال یال‌های سیناپسی پتانسیل فعال‌سازی گره را می‌سازد.**
- **تابع فعال‌سازی روی پتانسیل فعال‌سازی اعمال می‌شود و خروجی نورون تولید می‌شود.**

در شکل ۶ می‌توانید مدل نورون ارائه شده را به عنوان یک گراف جهتدار مشاهده کنید.



شکل ۶: مدل گراف جهتدار برای نورون [۱]

۲ـ۶ـ معماری شبکه عصبی

به نحوه قرار گرفتن گره در شبکه عصبی و اتصالات بین آن‌ها معماری شبکه عصبی می‌گویند. معماری شبکه‌های عصبی را به چند دسته‌ی اصلی تقسیم می‌کنیم:

۱. **شبکه های رو به جلو:** شبکه‌هایی که در ساختار گراف آن‌ها دور وجود ندارد.

- **شبکه های تک لایه:** شبکه‌های رو به جلویی که تنها شامل دو سطح از گره هستند سطح گره ورودی و سطح گره خروجی (نورون های خروجی).
- **شبکه های چند لایه:** شبکه‌هایی که حداقل دارای یک سطح بیشتر از شبکه‌های تک لایه هستند. (به لایه‌های میانی لایه‌های پنهان نورون نیز می‌گویند)

۲. **شبکه های درجریان:** شبکه‌هایی که در ساختار گراف آن‌ها دور وجود دارد.

۳ـ الگوهای اتصالات

در بخش قبل، دسته‌بندی انواع معماری‌های مختلف در شبکه‌های عصبی انجام شد. در این بخش می‌خواهیم این دسته‌بندی‌ها را به تفصیل بررسی کنیم.

در بین معماری‌های مختلف، ساده‌ترین‌شان شبکه‌های تک لایه‌ی رو به جلو و پر قدرت‌ترین‌شان شبکه‌های در جریان اند. اما به دلیل آن که برای این نوع شبکه‌ها به صورت کلی هنوز الگوریتم‌های یادگیری کارآمدی طراحی نشده، از آن‌ها کمتر استفاده می‌شود و در نتیجه پرکاربردترین معماری، همان شبکه‌های چند لایه‌ی رو به جلو اند که هم توانایی بسیار بالایی دارند و هم توانایی یادگیری بالا.

البته انواع مختلفی از شبکه‌های در جریان خاص ساخته شده‌اند که الگوریتم‌های یادگیری خوبی نیز دارند و در موضوعاتی که مورد استفاده قرار گرفته‌اند قدرت بالایی از خود نشان داده‌اند. اما هنوز اگر الگوریتم مناسبی برای حالت کلی شبکه‌های در جریان پیدا شود، می‌تواند باعث پیشرفت بزرگی بشود.

در واقع همانطور که در بخش ۳ـ۲ـ۱ـ نشان داده می‌شود، می‌توان با استفاده از نرون‌های بسیار ساده، تمام گیت‌های پایه‌ای یک رایانه را پیاده‌سازی کرد، پس قدرت محاسبه‌ای شبکه‌های عصبی مصنوعی حداقل به اندازه‌ی قدرت رایانه‌ها است (حداقل در تئوری) با این تفاوت که قدرت اصلی این شبکه‌ها در قابلیت یادگیری آن‌ها بدون کمک برنامه‌ریزی از قبل و دانستن مسئله‌های پیش رو است.

شبکه‌های عصبی را شاید بتوان همانند با سلول‌های بنیادی در نظر گرفت که می‌توانند بدون این که برای کاری اختصاصی شده باشند، آن‌ها را در محیط قرار دهیم و خود به خود، خودشان را با محیط تطبیق دهند و کار را به خوبی انجام دهند.

۳ـ۱ـ شبکه‌های تغذیه‌ی رو به جلو

ساده‌ترین نوع شبکه‌های عصبی، شبکه‌هایی اند که در ساختارشان دور وجود نداشته باشد. یعنی خروجی یک گره هرگز (بعد از یک یا چند مرحله) به خود آن گره برنگردد. در نتیجه اطلاعات همیشه از یک قسمت وارد شده و بعد از گذشتن از درون یک یا چند نورون مختلف به انتهای مسیر (گره‌های خروجی) می‌رسند. معمولا این شبکه ها را در حالت کلی به صورت شکل ۸ نشان می‌دهند.

اهمیت این معماری ساده در این است که با وجود سادگی، توانایی بالایی دارند و مهم‌تر از آن، برای آن‌ها الگوریتم‌های یادگیری کارآمد وجود دارد. در مقابل، معماری‌های پیچیده‌تری که جلوتر خواهیم دید، با وجود پتانسیل بالا نمی‌توانند از قدرت خود به حد کافی استفاده کنند (به جز در مواردی که برای یک دسته سوال اختصاصی شده‌اند).

۳ـ۲ـ شبکه‌های تک لایه

این نوع از شبکه‌های رو به جلو همانطور که از اسم‌شان مشخص است، فقط یک لایه نورون دارند (البته اگر گره‌های ورودی را نورون فرض نکنیم) و از آن‌ها با نام پرسپترون[۸] یاد می‌شود. این شبکه‌ها در سال ۱۹۶۰ توسط فرانک روزنبلت[۹] در دنباله‌ای از مقالات و یک کتاب[۱۰] بررسی شده و توسعه داده شدند. پس از انتشار این کتاب، شبکه‌های تک لایه محبوبیت زیادی به دست‌آوردند و نویسنده محبوبیت جهانی به دست آورد. پس از آن تحقیقات زیادی بر روی این شبکه‌ها انجام شد و محققان اکثرا انتظارات بسیار زیادی از آن‌ها داشتند تا این که مینسکی[۱۱] و پیرت[۱۲] در کتاب‌شان با نام پرسپترون [۱۳] در سال ۱۹۶۹ با نشان دادن محدودیت‌های این نوع شبکه‌ها، از شور و هیجان این تحقیقات به طور قابل ملاحظه‌ای کاستند تا حدی که تا سال ۱۹۸۰ تقریبا تحقیق بر روی این شبکه‌ها متوقف شد.[۲]

یک مثال قابل توجه از بزرگنمایی‌هایی که در مورد توانایی پرسپترون‌ها قبل از انتشار مقاله‌ی مینسکی و پیرت انجام گرفت، پروژه‌ای بود که در آن تعدادی عکس که در هر کدام دقیقا یک تراکتور یا یک تانک (در محیط‌های مختلف) قرار داشت به یک شبکه‌ی عصبی داده شد و هدف این بود که این شبکه بتواند در

[۸] perceptron
[۹] Frank Rosenblatt
[۱۰] Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan Books, 1962
[۱۱] Marvin Minsky
[۱۲] Seymour Paper
[۱۳] Perceptrons: an introduction to computational geometry, 1969

نهایت تشخیص بدهد که وسیله‌ی موجود در عکس تانک بوده یا تراکتور. حتی در بعضی از این عکس‌ها قسمتی از وسیله توسط درختان جنگلی پوشیده شده بود و وسیله به طور کامل دیده نمی‌شد. این پروژه در ابتدا نتایج بسیار موفقیت آمیزی داشت و به خوبی می‌توانست بین دو وسیله تشخیص درست بدهد. اما در نهایت مشخص شد که دلیل نتایج درست این شبکه این بوده که عکس تراکتورها در یک روز آفتابی و عکس تانک‌ها در یک روز ابری گرفته شده بود و تنها کاری که شبکه انجام می‌داد تعیین مقدار روشنایی تصویر و تصمیم‌گیری بر اساس آن بود، کاری که برای چشم انسان (حداقل با دقت بالا) کار ساده‌ای نیست، اما برای نورون ها به سادگی قابل انجام است. اینگونه اتفاقات می‌توانند یک الگوریتم را به سرعت بدنام کنند. [۳]

ساده‌ترین مثالی که شبکه‌های تک لایه در شبیه‌سازی‌شان عاجز اند، گیت منطقی xor است. البته این تنها مثال نیست و فقط نماینده‌ی گروه بزرگی از توابع است که نورون‌ها قابلیت تولید خروجی همانندشان را ندارند، اما خبر خوب این است که این مشکل با اضافه کردن تعداد لایه‌ها حل می‌شود و همین گیت xor با دو لایه نورون به سادگی قابل پیاده‌سازی است.

یک نکته‌ي مهم درتحلیل و بررسی رفتار شبکه‌های تک لایه این است که اگر دقت کنید، هر کدام از نورون‌ها کاملا به طور مستقل از دیگر نورون‌ها رفتار می‌کند چون مقادیر ورودی را که نمی‌توانیم تغییری بدهیم و بین نورون‌های سطح اول هم هیچ ارتباطی وجود ندارد. پس برای مثال هر شبکه‌ی تک لایه با k نورون را می‌توان k تا شبکه با فقط یک نورون در نظر گرفت که البته ورودی تمام‌شان را یکی می‌دهیم. این موضوع باعث می‌شود که تحلیل این شبکه‌ها بسیار ساده باشد و الگوریتم یادگیری کارایی برای آن‌ها طراحی شود. البته این موضوع باعث می‌شود که به سادگی ببینیم که این نوع شبکه‌ها توانایی بالایی در حل مسائل مختلف ندارند و با بزرگ‌تر کردن‌شان بر قدرت‌شان افزوده نمی‌شود.

۳ـ۲ـ۱ـ مثال‌هایی از قدرت تک نورون

برای نشان دادن حداقل توانایی های یک نورون ساده، با استفاده از آن‌ها گیت‌های منطقی پایه‌ای رایانه و دو مثال پیشرفته‌تر را طراحی می‌کنیم. برای این کار فرض کنید که تابع فعال‌سازی تمام نورون‌های این بخش و بخش بعد از نوع تابع Threshold (که در بخش ۲ـ۴ـ تعریف شده) است:

- **گیت not:** برای ساختن این گیت باید یک ورودی داشته باشیم که ضریب آن را برابر -۱ قرار می‌دهیم و مقدار عدد ثابت (bias) را برابر ۰ قرار می‌دهیم و یک گیت not ساخته می‌شود.

- **گیت and:** برای ساختن این گیت باید دو ورودی داشته باشیم که ضریب هر کدام را برابر ۱ قرار می‌دهیم و مقدار عدد ثابت را برابر ۱٫۵ قرار می‌دهیم. حال مقدار خروجی فقط در صورتی یک است که هر دو ورودی یک باشند.

- **گیت or:** همانند and دو ورودی داریم با ضریب ۱ و فقط مقدار عدد ثابت را برابر ۰٫۵ قرار می‌دهیم. حال اگر هر کدام را یک کنیم، جواب یک می‌شود. (در شکل ۷ هر دو گیت and و or نشان داده شده.)

- **تابع اکثریت** یعنی این که n ورودی داشته باشیم و بگوید که اکثریت آن‌ها صفر بوده‌اند یا یک: برای این تابع هم n ورودی داریم که ضریب هر کدام یک است و مقدار عدد ثابت برابر $-\frac{n}{۲}$ است.

- **فلیپ‌فلاپ**[۱۴]: ساخت این قطعه به داشتن دور در گراف ساختار نورون‌ها دارد و در نتیجه فقط در شبکه‌های در جریان (بخش ۳ـ۴ـ) قابل پیاده‌سازی است. اما برای ساخت آن فقط کافی است که خروجی نورون به عنوان ورودی دوباره به خودش داده شود (با ضریب ۱) و دو سیگنال ورودی set با ضریب ۱ و reset با ضریب منفی $۲-$ (البته بسته به استفاده می‌تواند $۱-$ هم باشد) و عدد ثابت (bias) هم برابر $۱-$ باشد. حال تا وقتی که سیگنالی نیاید، همیشه مقدار خروجی نورون ثابت می‌ماند و با آمدن سیگنال هم مقدار آن تعیین می‌شود.

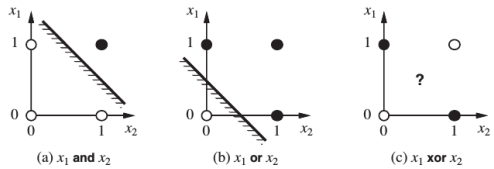
۳ـ۲ـ۲ـ ناتوانی‌های تک نورون

برای این که بگوییم یک نورون‌ها نمی‌تواند چه کارهایی را انجام دهد، ساده‌تر است که نشان دهیم چه کارهایی می‌تواند بکند و هر چه در آن قالب نگنجد، کارهایی است که نمی‌تواند انجام دهد. (در این بخش فقط با توابع فعال‌سازی Treshold کار می‌کنیم)

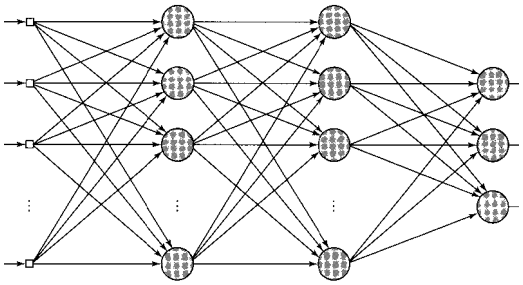
فرض کنید نورونی که می‌خواهیم بررسی کنیم n یال ورودی دارد. حال یک فضای n بعدی را در نظر بگیرید که به ازای هر یال ورودی یک بعد دارد. حال هر کدام از نقطه‌های این فضا n بعدی یک حالت از مقادیر ورودی نورون را نشان می‌دهد. حال کاری که نورون ما باید انجام بدهد این است که به ازای هر نقطه، مشخص کند که خروجی‌اش باید صفر باشد یا یک. حال با مقدار دهی به وزن‌ها (و مقدار ثابت (biass)) ما در واقع یک ابر صفحه‌ی n بعدی انتخاب می‌کنیم و به نقاطی که در یک طرف صفحه باشند مقدار یک و باقی نقاط مقدار صفر را اختصاص می‌دهیم.[۳]

با استفاده از این تعریف متوجه می‌شویم که یک نورون فقط در زمانی می‌تواند یک تابع با خروجی صفر و یک را تولید کند که

[۱۴] flip-flop



شکل ۷: جدایی ناپذیری خطی xor (کتاب راسل[۴، ص ـ ۷۳۰])



شکل ۸: معماری شبکه‌های چند لایه [۱]

نقاط آن (مثلا فقط نقاطی که خروجی‌شان یک است) خاصیت جدایی‌پذیری خطی[۱۵] داشته باشند.

ساده‌ترین تابعی که چنین خاصیتی را ندارد، تابع xor است که این موضوع در شکل ۷ به خوبی نشان داده شده است.

۳ـ۳ـ شبکه‌های چند لایه

این شبکه‌ها که حالت کلی شبکه‌های رو به جلو اند، امروزه مدل استاندارد مورد استفاده برای بیشتر مسائل تشخیص الگو با کمک یادگیری با ناظر اند. با ظهور و استفاده از این شبکه‌ها در الگوریتم‌های شبکه‌های عصبی، جان تازه‌ای در تحقیقات در این زمینه دمیده شد. این شبکه‌ها محدودیت‌های شبکه‌های تک لایه در تولید فقط بعضی از انواع توابع خروجی را ندارند.

یکی از مشکلاتی که شبکه‌های تک لایه با آن روبرو بودند این بود که برای این که بهتر کار کنند، نیاز بود که ابتدا از ورودی‌ها چند خاصیت کلی استخراج شوند و سپس با استفاده از این خاصیت‌ها پرسپترون‌ها کار تصمیم گیری را انجام بدهند. اما با استفاده از شبکه‌های چند لایه می‌توانیم ابتدا حتی بدون دانستن خروجی‌ها (بدون ناظر) مقدار زیادی خواص را فقط با داشتن ورودی‌های مختلف از آن‌ها استخراج کنیم و سپس با یا بدون داشتن خروجی‌ها شروع به تصمیم‌گیری کنیم. به همین دلیل این الگوریتم‌ها برای داده‌کاوی[۱۶] در مجموعه‌ی عظیمی از داده‌ها حتی وقتی که نمی‌دانیم دنبال چه چیزی می‌گردیم و برای مثال فقط می‌خواهیم دنبال بی‌قاعدگی‌ها بگردیم بسیار مناسب هستند. برای اطلاعات بیشتر و کاربردهای عملی این شبکه‌ها می‌توانید به ابزار متن‌باز وکا[۱۷] مراجعه‌کنید.

در معماری این شبکه‌ها تعدادی لایه نرون وجود دارد که نه گره ورودی‌اند و نه گره خروجی. به این لایه‌ها در اصطلاح، لایه‌های پنهان[۱۸] می‌گویند. اهمیت این لایه‌ها در این است که این لایه‌ها تنها چیزی‌اند که شبکه‌های چند لایه را از شبکه‌های تک لایه جدا می‌کنند، پس قدرت این شبکه‌ها بر دوش این لایه‌ها است. اما وجود این لایه‌ها مشکلاتی را برای الگوریتم‌های یادگیری ایجاد می‌کند.

در ادامه بعضی از این مشکلات را بررسی کرده و راه‌حل‌های آن‌ها را ارائه می‌دهیم.

۳ـ۳ـ۱ـ بیش‌برازش

بیش‌برازش[۱۹] به پدیده‌ی نامطلوبی در مدل‌های آماری گفته می‌شود که در آن درجه‌ی آزادی مدل بسیار بیشتر از درجه‌ی آزادی واقعی انتخاب شده و در نتیجه اگرچه مدل روی داده‌های استفاده شده برای یادگیری بسیار خوب نتیجه می‌دهد، اما بر روی داده‌های جدید دارای خطای زیاد است.[۵] برای مثال وقتی که مقدار یک چند جمله‌ای درجه‌ی دو را با یک چند جمله‌ای درجه‌ی سه تخمین بزنیم.

همانند تمام مدل‌های آماری، شبکه‌های چند لایه نیز وقتی که تعداد پارامترهای مسئله (همان تعداد لایه‌ها و اندازه‌ی کلی شبکه) بیش از حد باشد، به مشکل بیش‌برازش بر می‌خوریم. در این حالت حتی شبکه‌ی ما می‌تواند به صورتی تمام ورودی‌های داده‌شده را در خود ذخیره کند و بر روی آن‌ها درست جواب بدهد اما لزوما این نتایج را بر روی ورودی‌های جدید تعمیم ندهد.

به وضوح اگر اندازه‌ی شبکه بسیار کوچک باشد نیز مشکلات دیگری[۲۰] پیش می‌آید و نتایج دقت کافی را ندارند.

اگر قرار باشد در شبکه موجود تمام گره‌ها به یکدیگر وصل باشند، تنها موضوع باقی‌مانده، اندازه‌ی شبکه است. یکی از روش‌های ساده برای حل این مشکل این است که ابتدا چند ساختار را بررسی کنیم و کوچک‌ترین ساختاری را انتخاب کنیم که نتایج قابل قبولی دارد. روش‌های دیگر استفاده از روش اعتبارسنجی متقابل[۲۱] است. یک روش برای شبکه‌های تماما متصل نیستند، الگوریتم صدمه‌ی مغزی بهینه[۲۲] است که ابتدا از یک گراف کامل شروع می‌کند و سپس بعضی از ارتباطات آن‌را حذف می‌کند. [۴، ص ـ ۷۳۷]

[۱۵] Linear separability
[۱۶] DataMining
[۱۷] http://www.cs.waikato.ac.nz/ml/weka/
[۱۸] hidden layers
[۱۹] overfitting
[۲۰] underfittin
[۲۱] cross-validation
[۲۲] optimal brain damage

۳ـ۳ـ۲ـ انتشار رو به عقب

یکی از ابتدایی ترین مشکلات شبکه‌های چند لایه این است که نیاز به الگوریتم کارایی برای انجام یادگیری در لایه‌های نهفته داریم. به این دلیل که این لایه‌ها به طور مستقیم به خروجی‌ها وصل نیستند و نمی‌توانیم به طور مستقیم خروجی‌شان را بررسی کرده و بر اساس آن تغییرات لازم را انجام بدهیم. یکی از اولین و پرکاربردترین روش‌های ارائه شده، الگوریتم انتشار رو به عقب[۲۳] است. در این روش مقدار خطای لایه‌های عقب‌تر بر اساس خطای لایه‌های جلوتر به دست می‌آید و به اصطلاح خطا رو به عقب انتشار پیدا می‌کند. یک پیش‌نیاز برای انجام این روش این است که توابع فعال‌سازی نورون‌ها توابعی پیوسته باشند. برای مثال می‌توان از تابع فعال‌سازی Sigmoid (بخش ۲ـ۴ـ) استفاده کرد.

برای بهتر فهمیدن روش این الگوریتم، در نظر بگیرید که خروجی نورون‌های لایه‌های جلوتر یک تابع بر حسب خروجی نورون‌های لایه‌های مرحله‌ی قبل است. حال با در دست داشتن مقدار خطای تابع می‌توانیم مقدار تاثیر پارامترهای ورودی تابع را در خطا محاسبه کنیم. با انجام این کار و انتشار مقدار خطای به دست آمده به مراحل قبل، مقدار خطای آن‌ها نیز به دست می‌آید.[۳]

یک نکته‌ی مهم در مورد درستی این روش این است که شاید این روش بهترین روش ممکن نباشد، یک الگوریتم کارا هم نظر پردازشی و هم از نظر نتایج به دست آمده است.

اما هنوز این روش نمی‌تواند به عنوان یک روش یادگیری کامل بر روی تمام شبکه‌ها استفاده شود. به شبکه‌هایی که تعداد لایه‌های زیادی (در واقع تعداد لایه‌های پنهان زیاد) داشته باشند، شبکه‌های عمیق[۲۴] می‌گویند. در هنگام اجرای این الگوریتم بر روی این شبکه‌ها دیده می‌شود که یادگیری بعد از چند سطح دیگر کار خود را نمی‌تواند به خوبی انجام دهد. این مشکل به دلیل موضوعی به نام مسئله‌ی گرادیان محو شونده[۲۵] ایجاد می‌شود. یعنی سهم خطایی که برای لایه‌های قبلی محاسبه می‌کردیم، بعد از چند مرحله به سرعت از بین می‌رود و شاید حتی بعد از چهار مرحله، مقدار به دست آمده با تقریب خوبی برابر صفر باشد. به همین دلیل معمولا ابتدا برای لایه‌های اولیه از الگوریتم‌های یادگیری بدون ناظر استفاده می‌شود و سپس برای لایه‌های انتهایی از این الگوریتم یا الگوریتم‌های مشابه استفاده می‌شود.

[۲۳] back propagation
[۲۴] deep networks
[۲۵] Vanishing Gradient problem

۳ـ۴ـ شبکه‌های در جریان

این نوع شبکه‌ها بسیار قدرتمندتر از شبکه‌های رو به جلو اند. در کل در طراحی شبکه‌های در جریان[۲۶] می‌تواند دورهای جهت‌دار وجود داشته باشد، یعنی یک داده می‌تواند پس از گذشتن از چند مرحله دوباره به جای اولیه خود برگردد. به همین دلیل این شبکه‌ها توانایی نگه‌داری داده‌ها در طول زمان (عنصر حافظه همانند فلیپ‌فلاپ که در بخش ۳ـ۲ـ۱ـ نشان داده شد) را دارند. و همان طور که در بخش مذکور نشان داده شد، می‌توانند تمام اجزای پایه‌ای یک رایانه‌ی کامل را داشته باشند. پس اگر بتوانیم آن‌ها را تعلیم دهیم قدرت بسیار بالایی می‌توانند داشته باشند. اما به دلیل این که می‌توانند اشکال بسیار مختلف و پیچیده‌ای داشته باشند، الگوریتم‌های یادگیری مناسبی برای‌شان به دست نیامده و به همین دلیل در حال حاضر نمی‌توانیم از تمامی قدرت‌شان استفاده کنیم و طراحی چنین الگوریتمی بسیار مورد نیاز و سودمند است.

این شبکه‌ها طبیعی‌ترین روش برای مدل کردن داده‌های پشت سر هم (دنباله‌ی داده‌ها) هستند. معمولا در هنگام عمل یادگیری برای این شبکه‌ها آن‌ها را به صورت شبکه‌های چند لایه‌ی رو به جلو مدل می‌کنیم که در زمان عمق پیدا کرده اند. یعنی برای هر مرحله‌ی زمان، یک بار کل شبکه را قرار می‌دهیم و یال‌های شان را به جای وصل کردن به همان مرحله، به مرحله‌ی بعدی وصل می‌کنیم. یکی از خوبی‌های این روش این است که بااین کار می‌توانیم از الگوریتم‌های یادگیری ساخته شده برای شبکه‌های رو به جلو، در این شکبه‌ها نیز استفاده کنیم.[۳]

همچنین روش دیگری که بسیار مورد استفاده قرار می‌گیرد، استفاده از حالت‌های خاص توپولوژی نورون‌ها (در مقابل گراف کامل) بود که هر کدام به دلیل محدودیت‌های‌شان، تحلیل و بررسی و در نتیجه تعلیم‌شان ساده‌تر باشد.

۳ـ۵ـ مثال‌هایی از شبکه‌های در جریان

برای این که ببینید این الگوریتم‌ها چه کارهایی را می‌توانند انجام بدهند، چند مثال از فعالیت‌ها در این زمینه می‌آوریم:

در سال ۱۹۱۱ ایلیا ساتسکور[۲۷] یک نوع خاص از شبکه‌های در جریان را برای پیش‌بینی کاراکتر بعدی در یک دنباله از کلمات تعلیم داد. و سپس با استفاده از همین شبکه، یک متن کامل را از ابتدا تولید کرد. این متن شاید در نگاه کلی معنای خاصی نداشت اما با وجود این موضوع، همین که تمام کلمات تولید شده کلمات درست انگلیسی بودند و بسیاری از عبارات و حتی جملات آن معنای کامل و درست داشتند، خود نشان دهنده‌ی دقت و قدرت این الگوریتم بود.[۳]

[۲۶] recurrent
[۲۷] Ilya Sutskever

۳ـ۵ـ۱ـ شبکه‌های هاپفیلد

یک حالت خاص از شبکه‌های در جریان، شبکه‌های متقارن اند. در این شبکه‌ها یال‌ها به جای یک طرفه، دو طرفه اند. پس وزن در دو طرف یکسان است. جان هاپفیلد[۲۸] و دیگران متوجه شدند که تحلیل این نوع شبکه‌ها بسیار ساده‌تر از حالت کلی شبکه‌های در جریان است. در این شبکه‌ها علاوه بر دوطرفه بودن یال‌ها، یال به خود (طوقه) هم نداریم.

هاپفیلد و دیگران (همانند ماشین بلتزمن[۲۹] در بخش ۳ـ۵ـ۲ـ) بر اساس همین موضوع شبکه‌هایی را طراحی کردند. شبکه‌ی هاپفیلد معمولا به عنوان حافظه‌های تداعی‌گیر[۳۰] استفاده می‌شوند. ثابت می‌شود که این شبکه‌ها همیشه به یک کمینه‌ی موضعی[۳۱] همگرا می‌شوند. اما ضمانتی وجود ندارد که این کمینه‌ی موضعی همان جواب مسئله باشد. همچنین این شبکه‌ها به عنوان مدلی برای درک بهتر حافظه‌ی انسان نیز استفاده می‌شوند.[۶]

مسئله‌ی حافظه‌ی تداعی‌گر را می‌توان به نوعی ساده‌ترین مسئله برای نشان‌دادن روش محاسبه‌ی جمعی تصور کرد. صورت آن به صورت زیر است:

می‌خواهیم تعدادی الگو را در جایی ذخیره کنیم به صورتی که هر موقع الگوی جدیدی به ما داده شد، بتوانیم شبیه‌ترین الگو به آن را پیدا کنیم.

واضح است که می‌توانیم به ازای هر سوال، آن را با تمام مدل‌های داده شده مقایسه کنیم و سپس شبیه‌ترین را انتخاب کنیم، اما این روش کارآمد نیست. برای حل اینگونه سوال‌ها می‌توان از شبکه‌های هاپفیلد استفاده کرد. از کاربردهای این مسئله علاوه بر شبیه‌سازی حافظه‌ی انسان، می‌توان به تشخیص الگو و بازسازی عکس‌ها و بازیابی متون قدیمی که تا حدی از دست رفته اشاره کرد.[۷، ص‌ـ۱۱و۱۲]

۳ـ۵ـ۲ـ ماشین بلتزمن

هینتون[۳۲] و سجنوفسکی[۳۳] در سال ۱۹۸۵ یک قانون یادگیری جامع برای شبکه‌های احتمالاتی متقارن ابداع کرده(بخش ۵ـ۳ـ) و بر اساس آن ماشین بلتزمن را طراحی کردند.

ماشین بلتزمن همانند شبکه‌های هاپفیلد است با این تفاوت که می‌تواند لایه (یا لایه‌های) پنهان نیز داشته باشد. پس همانند شبکه‌های رو به جلو یک مشکل این است که بدون این که دانشی از مجموعه‌ی یادگیری داشته باشیم، بتوانیم ارتباطات مناسب را در این لایه‌های پنهان پیدا کنیم.

قابل خاطر نشان کردن است که الگوریتم اولیه‌ی بلتزمن به خاطر نیاز به محاسبه‌های طاقت‌فرسا بر روی متغیرهای احتمالاتی بسیار کند است اما مدل قطعی میدان میانگین[۳۴] سرعت یادگیری این ماشین را به طور چشم‌گیری افزایش می‌دهد.[۷، ص‌ـ۱۶۳]

ماشین بلتزمن معمولا به عنوان یک محاسبه‌گر واسطه استفاده می‌شود. برای مثال اگر آن را با عکس به عنوان ورودی تعلیم دهیم، می‌تواند برای کامل کردن یک عکس ناقص (که تکه‌ای از آن حذف شده) به کار رود.

۴ـ تعریف مسئله

جستجو و یافتن الگو درون داده‌ها را می‌توان یک مسئله‌ی کاملا پایه ای و پر کاربرد در علم، صنعت و به طور کلی در زندگی در نظر گرفت. برای مثال در قرن ۱۶ ام کپلربا توجه به اطلاعات زیاد مشاهداتی موجود متوجه الگوی حرکتی سیارات شد و یا مشاهدات طیف نشری اتم ها به پیدایش فیزیک کوانتوم ختم شد.

به طور کلی فیلد شناسایی الگوها به دنبال شناسایی خودکار نظم و قواعد درون داده ها توسط الگوریتم های کامپیوتری و پس از پیدا کردن این قواعد، به دنبال کار هایی همچون دسته بندی این داده‌ها در گروه‌های مختلف است. [۸، ص‌ـ۱]

شبکه‌های عصبی مصنوعی به خصوص در سال های اخیر به دلیل پیشرفت های حاصل در این زمینه به عنوان یک دسته از الگوریتم های قوی و پویای تشخیص الگو همیشه گزینه‌ی مناسبی برای حل این گونه مسائل بوده‌اند.

معیار هایی که با آن‌ها می توان این مسائل را تفکیک کرد متنوع اند، اما از میان آن‌ها می‌توان رایج ترین و مهم ترین آنها را نام‌برد که عبارت‌اند از، **نحوه‌ی یادگیری** که معمولا این مسائل را در دو فرم با نظارت و بدون نظارت دسته بندی می‌کند و **خواسته‌ی مسئله** که به طور عمده یکی از حالت های دسته بندی داده‌ها و یا پیشبینی داده های پیوسته است.

در ادامه هر یک از این معیار ها را بررسی می‌کنیم.

۴ـ۱ـ خواسته‌ی مسئله

الگویی که بر روی یک دسته از داده ها پیدا می‌شود را می‌توان یافتهی اصلی الگوریتم در نظر گرفت و با توجه به انواع الگوهایی که یک الگوریتم شناسایی می‌کند، می توان الگوریتم ها و مسائل متناظر آن ها را دسته بندی کرد.

این الگو در واقع یک تابع است که الگوریتم با بررسی داده های قدیمی پیدا می کند و از این تابع برای حدس زدن ویژگی‌های

[۲۸] John Hopfield
[۲۹] Boltzmann
[۳۰] Content-addressable memory
[۳۱] local minima
[۳۲] Hinton
[۳۳] Sejnowski
[۳۴] mean field

داده‌های جدید استفاده می‌کند. این تابع را در ادامه با $y(x)$ نشان می‌دهیم.

به طور کلی در یک مسئله‌ی یادگیری[۳۵] در صورتی که خواسته‌ی مسئله درون یابی[۳۶] یا برون یابی[۳۷] یک داده‌ی پیوسته باشد به آن یک مسئله‌ی رگراسیون[۳۸] می‌گویند و در صورتی که هدف پیشبینی یک کلاس گسسته و محدود برای داده های جدید باشد به آن کلاس‌بندی[۳۹] می‌گویند، یا به تفسیری دیگر از کتاب پترن ریکاگنیشن :

در صورتی که هدف یک فرایند، انتساب داده های ورودی به تعداد متناهی‌ای از کلاس‌های گسسته باشد به آن کار کلاس‌بندی می‌گوییم و اگر خروجی (یعنی خروجی تابع $y(x)$) شامل یک یا چند داده‌ی پیوسته باشد در این صورت به این کار رگراسیون می‌گوییم.[۸]

در نتیجه این دسته‌بندی بر اساس خروجی تابع y است.

۴ـ۲ـ نحوه‌ی یادگیری

فرض کنید یک مجموعه‌ی داده $X = x_1, x_2, ..., x_N$ در اخیار شما قرار گرفته و از شما می‌خواهند که با استفاده از آن یک روش و یا الگو ارائه دهید که در صورت مواجه با داده ی جدید $\widehat{x}$ با آن الگو بتوان یک ویژگی از $\widehat{x}$ به اسم $\widehat{t}$ را با تابع $y(\widehat{x})$ پیشبینی کرد.

همان طور که از جمله‌ی بالا معلوم است ویژگی خواسته شده توسط مسئله مبهم است و قبل از شروع به حل مسئله نیاز به رفع ابهام دارد.

در بعضی از مسائل علاوه بر X که مجموعه‌ی تعلیم[۴۰] نام دارد یک بردار $t = t_1, t_2, ..., t_N$ به نام بردار هدف[۴۱] از مقدار ویژگی مورد نظربه ازای ورودی های متناظر سوال به ما داده می‌شود، و هدف آن آست که با یافتن یک الگو در این ۲ مجموعه $y(x)$ را شکل دهیم و این ویژگی را برای ورودی هایی که $\widehat{t}$ به ما داده نشده با $y(\widehat{x})$ پیشبینی کنیم، به یادگیری این دسته از مسائل به علت راهنمایی هایی که در مرحله‌ی یادگیری به الگوریتم داده می‌شود، یادگیری با نظارت[۴۲] گفته می‌شود.

اما در بعضی دیگر از مسائل بردار t به الگوریتم داده نمی‌شود، با این که ممکن است در نگاه اول عجیب به نظر برسد که الگوریتم باید بدون داشتن هیچ ایده ای از خواسته‌ی مسئله به دنبال ویژگی‌ای در مسئله بگردد، اما در ادامه می‌بینیم در واقعت بسیار با این گونه مسائل روبه‌رو می‌شویم، برای مثال وقتی که برای اولین بار کسی با چند دسته از اشیاء جدید روبه‌رو می‌شود، بدون آن که کسی به او بگوید این اشیاء با هم تفاوت دارند، فرد از روی تفاوت هایی مانند شکل، اندازه و ... آن‌ها را در گروه هایی دسته بندی می‌کند و برای هر گروه مفهومی در ذهن خود درنظر می‌گیرد، از آن جا که این دسته از مسائل هیچ گونه راهنمایی‌ای در مرحله‌ی یادگیری دریافت نمی‌کنند، یادگیری این دسته مسائل را یادگیری بدون نظارت می‌نامند.

۵ـ یادگیری

پر اهمیت ترین ویژگی شبکه های عصبی، توانایی یادگیری و بهبود عملکرد آن‌ها به وسیله‌ی یادگیری است. در واقع اتصالات بین عصب‌ها و شیوه‌ی فعالیت آن‌ها، تابع $y(x)$ که در قسمت قبل مطرح شد را می‌سازند، پس در شبکه‌های عصبی مرحله‌ی یادگیری که مرحله‌ی تولید y است، با تعیین وزن اتصالات بین عصب ها انجام می‌شود. تعریف دقیق یادگیری در شبکه های عصبی را می توان به صورت زیر دانست:

یادگیری فرایندی است که در آن متغیرهای آزاد یک شبکه‌ی عصبی (یعنی وزن اتصالات و ...) در جریان تحریک شدن به وسیله‌ی محیطی که شبکه در آن قرار دارد مقدار می‌گیرند. نوع یادگیری، تغییراتی که در این متغیرها رخ می‌دهد را تعیین می‌کند.[۱، ص ـ ۵۰]

با توجه به تعریف بالا می‌توان گفت که یادگیری در شبکه‌های عصبی از دنباله‌ی وقایع زیر تشکیل شده:

۱. تحریک توسط یک محیط.

۲. تغییر متغیرهای آزاد بر اساس این تحریک.

۳. پاسخ دادن به صورتی جدید به محیط به خاطر تغییرات مرحله‌ی قبل.

همان طور که تعریف به آن اشاره کرد یادگیری انواعی دارد که روش تغییرات داخلی را تعیین می‌کنند. از آنجا که این تغییرات به صورت های متفاوتی می‌توانند شکل گیرند، می‌توان انواع زیادی از یادگیری را تولید کرد، چند نمونه از روش های یادگیری معروف را می‌توان روش های زیر دانست:

۱. تصحیح خطا

۲. یادگیری هبین

۳. یادگیری رقابتی

۴. یادگیری برمبنای حافظه

۵. یادگیری بتلزمن

---

[۳۵] Learning Problem
[۳۶] Interpolation
[۳۷] Extrapolation
[۳۸] Regression
[۳۹] Classification
[۴۰] Training set
[۴۱] Target vector
[۴۲] Supervised learning

در بخش های بعدی درباره‌ی هر یک از این روش های یادگیری توضیحی مختصر داده می‌شود، برای یادگیری بیشتر در این زمینه می‌توانید به منابع ذکر شده در قسمت پایانی رجوع کنید.

۵ـ۱ـ قواعد تصحیح خطا

در ساده ترین حالت این روش یادگیری می توانید فرض کنید یک عصبk داریم که خروجی آن قابل اندازه‌گیری است و آن را با $y(x)$ نشان می دهیم، برای این عصب یک مقدار مطلوب خروجی وجود دارد که آن را $d(x)$ می‌نامیم و مستقیما به ما داده می‌شود. در اثر مقایسه‌ی سیگنال خروجی با سیگنال مطلوب یک سیگنال خطا به دست می‌آید که برابر با مقدار زیر است:

$$e_k(n) = d_k(n) - y_k(n)$$

سیگنال خطای e یک مکانیزم کنترلی را فعال می‌کند که هدف آن اعمال یک سری تغییرات بر روی وزن اتصالات عصب ها به منظور بهبود دادن خروجی است. این تغییرات طراحی شده اند تا مرحله به مرحله فاصله‌ی $y(x)$ را از $d(x)$ کم کنند. در هر مرحله با انتخاب تغییراتی که مقدار تابع هزینه‌ی زیر را کمینه کند این کار را انجام می‌دهیم:

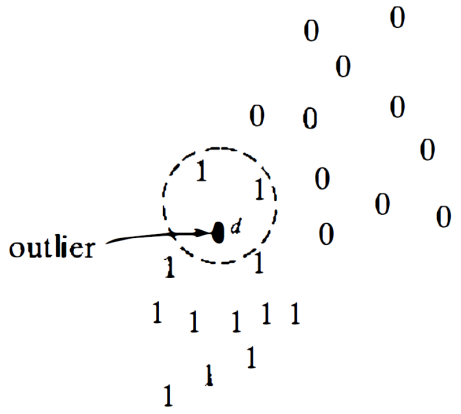
$$\xi(n) = \frac{۱}{۲} e_k^۲(n)$$

که در اینجا $e_k(n)$ نشان دهنده‌ی مقدار خطا در مرحله‌ی n ام است. می توان این تابع هزینه را تابع انرژی لحظه‌ای خطا نامید. این تغییرات مرحله به مرحله تا جایی ادامه می‌یابد که عصب k به یک وضعیت تعادل برسد، در این حالت فرایند متوقف می‌شود.می‌توان نشان داد که کمینه کردن این تابع هزینه معادل است با انتخاب یک بردار تغییرات $\Delta\omega$ به صورت زیر که مقدار ورودی ها و مقدار خطا متناسب است.

$$\Delta\omega_{k,j} = \eta e_k(n) x_j(n)$$

$$\omega_{k,j}(n+۱) = \omega_{k,j}(n) + \Delta\omega_{k,j}(n)$$

که $\omega_{k,j}(n)$ وزن اتصال بین عصب k و j در مرحله‌ی n است و η ضریب یادگیری شبکه است که مقدار آن سرعت فرایند یادگیری را تعیین می‌کند. برای اندازه‌گیری خطا لازم است که علاوه بر مقدار مطلوب به خود خروجی نیز دسترسی داشت پس k باید یک عصب خروجی باشد که همیشه ممکن نیست، به علاوه این روش به صورت موضعی عمل می‌کند، یعنی خطا تنها از روی عصب های مجاور به دست می‌آید و حالت های پیچیده تر را درنظر نمی‌گیرد.



شکل ۹: محدوده‌ی درون خطچین شامل دو نقطه متعلق به کلاس ۱ و یک داده‌ی پرت با مقدار ۰ است، الگوریتم k نزدیک ترین همسایه ($k = ۳$) مقدار به طور شهودی صحیح ۱ را به نقطه‌ی d نسبت می‌دهد، در حالی که d به داده‌ی پرت ۰ نزدیک تر است.(کتاب شبکه های عصبی[۱، ص‌ـ۵۵])

۵ـ۲ـ یادگیری بر مبنای حافظه

در یادگیری بر پایه‌ی حافظه، بیشتر تجربیات گزشته به صورت واضح در یک حافظه‌ی رده بندی شده به صورت ورودی‌ـ خروجی یعنی $\{(X_i, d_i)\}_{i=۱}^{N}$ ذخیره می‌شوند، که x و d به ترتیب نشان دهنده‌ی ورودی و خروجی مورد نظر است. فرض کنید مقدار خروجی یکی از مقدارهای ۰ یا ۱ باشد. در این صورت با دریافت $\widehat{x}$ (ورودی جدید) الگوریتم با بررسی داده‌های آزمایشی در همسایگی $\widehat{x}$ و آنالیز آن‌ها پاسخ خود را می‌دهد.

در تعریف بالا دو چیز نیاز به رفع ابهام دارد، یکی مفهوم همسایگی $\widehat{x}$ و دیگری روش اعمال قوانین یاگیری بر روی داده های آزمایشی در همسایگی $\widehat{x}$ است. برای مثال یک روش یادگیری ساده مبتنی بر حافظه قانون نزدیک ترین همسایه[۴۳] است. در این جا همسایگی $\widehat{x}$ ، آزمایشی است که در نزدیک ترین فاصله از آن قرار دارد یعنی آزمایشی که بردار ورودی آن تا بردار $\widehat{x}$ کمترین فاصله‌ی اقلیدسی را داراست:

$$\min_i d(x_i, \widehat{x}) = d(x_N^{'}, \widehat{x})$$

مقداری که به این داده یعنی $x_N^{'}$ نسبت دارد به عنوان مقدار خروجی $\widehat{x}$ گزارش می‌شود، می‌توان ثابت کرد در صورت برقرار بودن چند شرط بر روی داده ها، احتمال خطای این مقدار خروجی، حداکثر ۲ برابر احتمال خطای بیز است، یعنی حداکثر دو برابر احتمال خطای بهینه.

یک حالت دیگر از یادگیری‌های مبتنی بر حافظه که در واقع تعمیم روش قبل است، کلاس‌بندی k نزدیک ترین همسایه است.

[۴۳] Nearest neighbor rule

در این روش همسایگی $\widehat{x}$، k آزمایشی است که در نزدیک ترین فاصله از بردار ورودی قرار دارند، و الگوریتم با یافتن کلاسی که در این k آزمایش بیشتر از همه تکرار شده ، جواب مسئله را می‌دهد.

الگوریتم های دیگری نیز وجود دارند که بر پایه‌ی حافظه هستند اما ما به همین دو مورد بسنده می‌کنیم.

شکل ۹ تفاوت این دو روش مبنی بر حافظه را نشان می‌دهد.

۵ـ۳ـ بلتزمن (Boltzmann)

در سال ۱۹۸۶ هینتون و سجنوفسکی یک قانون یادگیری را برای شبکه‌های احتمالاتی متقارن ابداع کردند و نام آن را به خاطر توزیع بلتزمن به افتخار طراح آن نام گذاری کردند. به شبکه های عصبی‌ای که بر پایه‌ی این الگوریتم کار می‌کنند ماشین های بلتزمن (بخش ۳ـ۵ـ۲ـ) می‌گویند.

در یک ماشین بلتزمن هر عصب می‌تواند در ۲ حالت مختلف مثلا ۱+ یا ۱− قرار گیرد و برای کل سیستم می‌توان یک تابع انرژی E در نظر گرفت که مقدار آن با وضعیتی که عصب ها می‌گیرند تعیین می‌شود:

$$E = -\frac{۱}{۲}\sum_{j}\sum_{\substack{k \\ j\neq k}}\omega_{kj}x_k x_j$$

ماشین در هر مرحله با انتخاب یک عصب تصادفی و عوض کردن وضعیت آن عصب با احتمالی که تابع زیر توصیف می‌کند در دمای T تلاش می‌کند تا مقدار انرژی را به دمای تعادل برساند (البته که این دما کمیتی فیزیکی نیست و می‌توان آن را شبه دما نامید).

$$P(x_k \to -x_k) = \frac{۱}{۱ + \exp\left(-\Delta E_k/T\right)}$$

اینجا ΔE_k مقدار تغییرات انرژی بر اثر تغییر وضعیت عصب k را نشان می‌دهد.
عصب های ماشین بلتزمن را می‌توان به دو دسته‌ی پنهان و آشکار تقسیم کرد، این ماشین در دو حالت مختلف عمل می کند:

۱. حالتی که وضعیت عصب های آشکار از سوی محیط تحمیل می‌شود.

۲. حالتی که عصب های آشکار آزادانه تغییر می کنند.

که در هر دو این حالات وضعیت عصب های پنهان به صورت آزاد می‌تواند تغییر کند. فرض کنید $\rho^{+}_{i,j}$ مقدار هم بستگی وضعیت های دو عصب i و j و میانگین گرفته شده درتمام وضعیت‌های تعادل حالت اول است. و $\rho^{-}_{i,j}$ همین هم بستگی در حالت دوم است. تغییر وزن اتصال بین این دو عصب را با فرمول زیر تعیین می‌کنیم:

$$\Delta\omega_{k,j} = \eta(\rho^{+}_{k,j} - \rho^{-}_{k,j}), \quad j \neq k$$

که متغیر های بالا قبل تر تعریف شده اند.

۵ـ۴ـ هبین (Hebian)

فرضیه‌ی هب یکی از قدیمی ترین و مشهور ترین قانون های یادگیری را ارائه می‌کند:
وقتی که یک عصب A به اندازه کافی به عصب B نزدیک باشد که آن را تحریک کند و به طور مداوم در فعال کردن عصب B شرکت کند، فرایندی در یکی یا هر دو این عصب ها شکل می‌گیرد که تاثیر عصب A را بر روی عصب B افزایش می‌دهد. [۱، ص‌ـ ۵۱]
با عوض کردن جمله بندی این گزاره، دو قانون زیر را به دست می‌آوریم:

۱. اگر دو عصب در دو سمت یک سیناپس (یک اتصال) به طور هم‌زمان عمل کنند، در آن صورت قدرت آن سیناپس به دلخواهی افزایش می‌یابد.

۲. اگر دو عصب در دو سمت یک سیناپس (یک اتصال) به طور غیر هم زمان عمل کنند، در آن صورت قدرت آن سیناپس به دلخواهی کاهش می‌یابد. [۱، ص‌ـ۵۵]

به چنین سیناپسی سیناپس هبین می‌گویند، در واقع چنین اتصالی از یک فرایند فعال وابسته به زمان و به شدت موضعی برای بهبود کارایی عصب ها و بهینه سازی آنها استفاده می‌کند.
از تعریف اولیه، قانون دوم نوشته شده به دست نمی‌آید در واقع مدل های متفاوتی با توجه به قوانین هبین می‌توان ساخت که می‌توانند به صورت متفاوتی عمل کنند، برای مثال ممکن است تغییرات وزن اتصالات متناسب با حاصل ضرب وضعیت دو عصب تغییر کند که در این صورت سیناپس خاصیت کواریانسی پیدا می‌کند و یا این که تنها با هم‌زمانی ها تقویت شود و تضعیفی در کار نباشد که مشابه تعریف اولیه است.
به شباهت قوانین هبین با پدیده‌ی شرطی سازی در روانشناسی نیز می‌توان توجه کرد.

۵ـ۵ـ آموزش رقابتی

مشابه اسم این یادگیری، عصب های شبکه‌های عصبی در این روش با یکدیگر سر فعال شدن رقابت می‌کنند، برعکس روش های قبلی که در آن به طور هم‌زمان چند عصب از شبکه می‌توانستند فعالیت کنند، در این روش تنها عصب برنده فعال می‌شود، که این خاصیت این شبکه ها را برای کشف ویژگی های مهم آماری برای

دسته بندی داده ها مناسب می‌سازد. قواعد این روش را در سه عنصر می‌توان خلاصه کرد:

۱. یک مجموعه از عصب ها که کاملا مشابه اند، به جز در وزن اتصالات که به صورت تصادفی توزیع شده اند.

۲. یک محدودیت بر روی قدرت عصب ها.

۳. یک مکانیزم رقابت که عصب ها به وسیله‌ی آن بتوانند رقابت کنند و برنده تعیین شود.

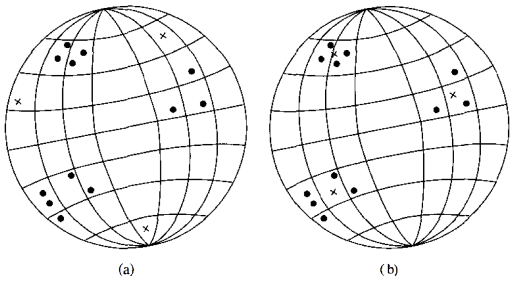
در ساده ترین حالت این شبکه می‌توان یک سطح از عصب های خروجی در نظر گرفت که همه‌ِی عصب های ورودی به همه‌ی آن‌ها اتصالاتی با وزن مثبت دارند (اتصالات بین عصب های خروجی نیز مشکلی ایجاد نمی‌کند).

برای این که عصب k بین عصب ها برنده شود باید میدان القایی[۴۴] آن به ازای ورودی $\widehat{x}$ در بین همه‌ی عصب ها بیشینه باشد، در نتیجه ورودی های عصب برنده به نحوی تغییر می‌کنند که عصب k با ورودی $\widehat{x}$ آموزش یابد و این عصب با این ورودی راحت تر تحریک شود، برای این کار بردار وزن اتصالات عصب k به سمت ورودی $\widehat{x}$ شیفت داده می‌شود، یا به عبارتی:

$$\Delta\omega_{k,j} = \begin{cases} \eta(x_j - \omega_{k,j}) & \text{اگر عصب } k \text{ مسابقه را ببرد} \\ ۰ & \text{اگر عصب } k \text{ مسابقه را ببازد} \end{cases}$$

که ضریب η همان ضریب یادگیری است. تعبیر شهودی فرایند ذکر شده این است که در صورت خوشه ای بودن داده ها در چند همسایگی، برای دسته بندی این داده ها به وسیله‌ی شبکه‌ی عصبی بالا در گروه‌های جدا از هم، می خواهیم که به هر گروه یک عصب نسبت دهیم که در صورتی که ورودی درون یکی از آن گروه ها فعال شود، عصب متناظر با آن گروه شروع به فعالیت کند. برای این کار بردار ورودی عصب ها را رندم در فضای حالات پراکنده می‌کنیم و در صورتی که بردار ورودی یک عصب به ورودی یک آزمایش نزدیک تر باشد(و در نتیجه بیشتر از بقیه‌ی عصب ها تحریک شود)، آن عصب را متناظر با گروه آن ورودی در نظر می‌گیریم و بردار آن را به بردار این ورودی نزدیک می‌کنیم. بعد از چندین مرحله اجرای این آزمایش به حالتی می‌رسیم که بردار هر عصب در میان گروه متناظر آن قرار دارد و در مراحل بعدی این عصب اعضای آن گروه را شناسایی می‌کند، از آن جا که بردار های ورودی در ابتدا تصادفی پراکنده شده بودند، به صورت احتمالی هر گروه یک عصب درون خود دارد، و از آنجا که هر آزمایش یک برنده دارد، در صورتی که یک گروه عصب متناظر خود را پیدا کند، از آن پس عصب برنده‌ی خود را خواهد داشت و بردار عصب های دیگر را به سمت خود نمی‌کشاند.

[۴۴] Local induced field



شکل ۱۰: فرایند یادگیری عصب ها در شبکه های رقابتی: شکل سمت چپ قبل از تحریک شدن توسط داده ها، شکل سمت راست بعد از تحریک شدن. هر یک از علامت های ضربدر نشان دهنده ی مکان یکی از بردارهای وزن اتصالات است، هر یک از نقطه ها نشان دهنده ی بردار ورودی یک داده است.(کتاب شبکه های عصبی[۱]، ص‌ـ۶۰])

شکل ۱۰ این فرایند را نشان می‌دهد.

۶ـ کاربردهای مهم

در چند دهه‌ی اخیر بسیاری از فعالیت هایی که پیش از این تنها به وسیله‌ی انسان قابل انجام بوده، به وسیله‌ی کامپیوتر دست یافتنی شده است. علت این پیشرفت را می‌توان پیدایش پردازنده های قوی تر و همچنین الگوریتم های جدید ماشین‌لرنینگ دانست. به عنوان یکی از این الگوریتم ها شبکه‌های عصبی نیز نقش موثری در این فرایند داشته‌اند. مواردی که در ادامه مطرح می‌شود به طور کلی کاربرد هایی هستند که به وسیله‌ی شبکه های عصبی و یا سایر ماشین های یادگیری آماری ممکن شده اند.

۶ـ۱ـ تشخیص الگو، صدا و تصویر

امروزه پیشرفت ها در این زمینه تا حدی بوده که حتی بعضی از تلفن های همراه هم دارای امکاناتی همچون دستیار های صوتی و یا تشخیص چهره هستند، گرچه در ابتدا موفقیت های ماشین های آماری در این زمینه بسیار بیشتر از شبکه‌های عصبی بود ولی با پیدایش شبکه های عمیق و پیشرفت شبکه‌های عصبی، این شبکه‌ها از ماشین های آماری پیشی گرفتند.

در این مورد معرفی چند مورد مناسب به نظر می‌رسد:

۱. در الگوریتم‌های داده‌کاوی برای پیدا کردن الگوها و گشتن به دنبال بی‌قاعدگی‌ها به طور گسترده از شبکه‌های چند لایه (بخش ۳ـ۳ـ) استفاده می‌شود.

۲. برای پیدا کردن الگو مشابه در یک بانک از الگوها می‌توان از شبکه‌های هاپفیلد (بخش ۳ـ۵ـ۱ـ) استفاده کرد.

۳. برای بازسازی عکس‌های آسیب‌دیده یا ناقص و یا بازیابی

اطلاعات متون خطی که تا حدی از دست رفته‌اند می‌توانیم از شبکه‌های بلتزمن (بخش ۳ـ۵ـ۲ـ) یاری بجوییم.

۶ـ۲ـ خوشه‌سازی و دسته‌بندی

وقتی حجم داده‌ها زیاد و یا تعداد ابعاد ورودی بیش ۳ یا ۴ بعد می‌شود ادراک انسان دچار مشکل شده و توانایی خود را در تجسم داده از دست می‌دهد، این مشکل برای کامپیوتر نیز به دلیل افزایش محاسبات به صورت نمایی با افزایش ابعاد ورودی وجود دارد اما شدت آن کم‌تر است.

الگوریتم‌های شبکه‌های عصبی تا به امروز در این کار بسیار موفق عمل کرده‌اند. در بعضی از این الگوریتم‌ها حتی نیاز به دانستن دسته‌بندی‌ها هم نیست (یادگیری بدون نظارت) و شبکه‌های عصبی تنها با داشتن ورودی‌های محتلف می‌توانند تا حد خوبی خصوصیات و ویژگی‌های مختلف داده‌ها را از یکدیگر تشخیص بدهند.

به عنوان مثال در این باره می‌توان شبکه‌های LSTM[۴۵] را نام برد که در سال ۲۰۰۹ برای تشخیص متن دست‌نوشت به کار رفت و مسابقه‌ی ICDAR در همین زمینه را در سال مذکور از آن خود کرد.

۶ـ۲ـ۱ـ استخراج اطلاعات

انسان ها در برابر حجم زیاد داده ها از ماشین ها عقب می مانند، علاوه بر این، گاهی ماشین ها نظمی در داده ها پیدا می‌کنند که انسان به طور شهودی قابلیت درک آن را ندارد. که باز هم می‌توان استفاده‌ی این الگوریتم‌ها در زمینه‌ی داده‌کاوی را خاطر نشان کرد.

۶ـ۲ـ۲ـ تحلیل اطلاعات

کامپیوتر ها در این زمینه هم ثابت کردند که در بعضی از موارد از انسان ها سریع ترند، بسیاری از شرکت های اقتصادی بر پایه‌ی این برنامه ها سرمایه‌گذاری های خود را انجام می‌دهند.

۶ـ۳ـ بهینه‌سازی مسائل پیچیده

مسائل NP دسته ای کاربردی از مسائل هستند که تا کنون الگوریتم ای قطعی برای حل آن ها در زمان مناسب ارائه نشده است. اما شبکه های عصبی و سایر ماشین‌های یادگیری می‌توانند آن ها را در زمان خیلی خوبی با اطمینان بالایی حل کنند. و شاید حتی این دسته از مسئله ها را بتوان علت این دانست که چرا الگوریتم های قطعی توانایی رقابت با مغز انسان را ندارند.

۶ـ۴ـ سایر کاربردها

چند نمونه دیگر از موارد کاربردی شبکه های عصبی:

۱. تقریب تابع‌ها: تقریب زند توابعی که محاسبه ی آن ها به راحتی امکان پذیر نیست.

۲. پیش‌بینی و حدس: پیشبینی وضع هوا، بازار، حدس زدن نتیجه‌ی یک رخداد و ....

البته تنوع کاربردی شبکه های عصبی بیشتر از مواردی که در اینجا بیان شده است.

[۴۵] Long Short-term Memory

مراجع

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., New Jersey, Prentice-Hall, 1999.

[2] Wikipedia contributors, "Frank Rosenblatt" *Wikipedia, The Free Encyclopedia*, [online], Available: *https://en.wikipedia.org/wiki/Frank_Rosenblatt*

[3] G. Hinton, *Neural Networks and Machine Learning [lecture notes]*, Retrieved from coursera.org, 2012

[4] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., New Jersey, Prentice-Hall, 2010

[5] Wikipedia contributors, "Overfitting" *Wikipedia, The Free Encyclopedia*, [online], Available: *https://en.wikipedia.org/wiki/Overfitting*

[6] Wikipedia contributors, "Hopfield network" *Wikipedia, The Free Encyclopedia*, [online], Available: *https://en.wikipedia.org/wiki/Hopfield_network*

[7] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the theory of Neural Computation*, Lecture notes Volume I, California, Addison-Wesley publishing company, 1990.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., New York, Springer, 2006.

[9] A. Ng, *Machine Learning [lecture notes]*. Retrieved from coursera.org, 2014