

# مروری بر الگوریتم‌های شبکه‌ی عصبی

فرزاد عبدالحسینی، سید سبحان میریوسفی، هومن هاشمی  
دانشگاه صنعتی شریف، دانشکده مهندسی کامپیوتر  
{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 نا حدى بررسی شده و در انتها بعضی از کاربردهای مهم عملی این شبکه‌ها نشان داده شده است. کلید واژه‌ها - الگوریتم، هوش مصنوعی، شبکه‌های عصبی.

## ۱ - مقدمه

## ۲ - آشنایی و مفاهیم اولیه

در اینجا ابتدا یک تعریف ارائه می‌کنیم [۱] و سپس در بخش‌های بعد به توضیح آن می‌پردازیم:

شبکه عصبی مصنوعی یک پردازنده توزیع شده موازی و گسترده است و که از واحد‌های پردازشی ساده ساخته شده است که می‌تواند دانش اکتسابی را در خود ذخیره کند و در آینده از آن‌ها 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 واحد‌های پردازشی (نورون‌ها) که به آن وزن سیناپسی می‌گوییم برای ذخیره اطلاعات استفاده می‌شود.

### ۱-۲ - 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 اینکه قابلیت یادگیری است اینکه می‌تواند با توجه به محیط یک سری اطلاعات و تجارب کسب کند و از این تجارب در تصمیم‌گیری‌های بعدی استفاده کند به این ویژگی کلیت بخشی<sup>۱</sup> می‌گوییم. این ویژگی‌ها باعث می‌شود مسائلی که در حال حاضر برای کامپیوترهای مرسوم دست‌نیافتی است توسط این شبکه‌ها قابل حل باشد البته باید توجه داشت که گونه‌های خاصی از مسائل هستند که توسط این شبکه‌های عصبی به طور بهینه قابل حل هستند و پردازنده‌های مرسوم در بعضی موارد بهتر

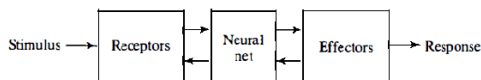
کار بر روی شبکه‌های عصبی مصنوعی یا به اختصار شبکه‌های عصبی از جایی آغاز شد که دانشمندان به این مهم دست یافتند که سیستم پردازش مغز انسان بسیار متفاوت با سیستم‌های کامپیوتری دیجیتال مرسوم می‌باشد. مغز انسان از یک ساختار بسیار پیچیده، غیر خطی و موازی بهره می‌برد و همچنین قابلیت بهبود و ارتقای خود را نیز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

برای مثال قدرت بینایی و بصری انسان را در نظر بگیرید که یک نوع پردازش اطلاعات است. ما اطلاعات رَو از محیط بیرون توسط حسگر پیچیده چشم دریافت می‌کنیم آن‌ها را تجزیه و تحلیل می‌کنیم تا بتوانیم چیزهایی 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 فرآیند‌های بسیار ساده تر از این برای کامپیوترهای 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 انعکاس آن توسط طعمه می‌تواند اطلاعاتی نظیر سرعت نسبی طعمه، اندازه طعمه و ... بدست آورد. تمام این پردازش‌های پیچیده در مغز کوچک خفاش که به اندازه‌ی یک آلو است انجام می‌پذیرد.

اما مغز انسان یا خفاش چگونه این کار‌ها را انجام می‌دهد؟

<sup>۱</sup>Generalization



شکل ۱: ساختار سه بخشی شبکه عصبی انسان

- **آزمایشات زیستی:**<sup>۶</sup> از آن جایی که شبکه عصبی تا حد خوبی میتواند شبکه های عصبی واقعی را تداعی کند از آن میتواند در آزمایشات و پژوهش های زیستی استفاده کرد.

## ۲-۲- بررسی مغز انسان

از آن جایی که شبکه عصبی برگرفته از مغز انسان است ابتدا به بررسی آن 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 نورون ها است و محل تصمیم گیری میباشد. بخش اول یا همان گیرنده ها اطلاعات را از محیط گرفته و به سیگنال های قابل فهم برای شبکه عصبی تبدیل میکند و بخش آخر نیز دستورات را از شبکه عصبی گرفته و واکنش مورد نظر را در محیط انجام میدهد. در سال ۱۹۱۱ ساختار نورونی<sup>۷</sup> برای مغز معرفی شد. نورون ها ۵ الی ۶ مرتبه توانی از سیلیکون کند تر هستند. اتفاقات در چپ های سیلیکونی هر ۱ نانوثانیه اتفاق میافتد و این در حالی است که در نورون هر ۱ میلی ثانیه. ولی مغز انسان این سرعت کم نورون ها را با تعداد بسیار زیاد آن ها و اتصالات (سیناپس ها) بسیار زیاده جبران کرده است. تخمین زده میشود که تعداد نورون ها حدود ۱۰ میلیارد و تعداد اتصالات بین آن ها حدود ۶۰ تریلیارد است. از طرفی دیگر مصرف انرژی مغز انسان در مقایسه با چپ 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 میکنید. در اینجا به تشریح بخش های مختلف می پردازیم:

- **سیناپس ها یا اتصالات** هر کدام به همراه یک عدد که به آن وزن سیناپسی میگوییم مشخص شده اند 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.

عمل میکنند. باید توجه داشت بخش های مربوط از یک مسئله را توسط این شبکه ها حل کنیم. همچنین تا ساختن شبکه عصبی که به طور کامل مغز انسان یا هر شبکه عصبی طبیعی دیگر را تداعی کند راه طولانی ای در پیش داریم. حالا بعضی از مزایا و توانایی های شبکه های عصبی را ذکر میکنیم:

- **توانایی غیرخطی بودن:**<sup>۲</sup> واحد های سازنده شبکه عصبی یا همان نورون ها میتوانند غیرخطی باشند و در نتیجه کل شبکه عصبی غیرخطی میشود این ویژگی بسیار مهم است زیرا بسیاری از مسائل ذاتا غیرخطی هستند مانند پردازش گفتار.

- **نگاشت ورودی-خروجی:**<sup>۳</sup> 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 این ها شبکه عصبی را آموزش دهیم در اینصورت شبکه عصبی وزن های سیناپسی خود را به گونه ای تغییر میدهد که جواب تاحدممکن به خروجی مورد نظر نزدیک شود در اینصورت به گونه ای توانستیم بدون هیچ دانسته قبلی بین ورودی و خروجی های درست نگاشت برقرار کنیم حال میتوانیم از شبکه عصبی برای بدست آوردن خروجی مورد نظر برای ورودی های دیگر استفاده کنیم این تکنیک در مسائل الگویی بسیار استفاده میشود.

- **تطبیق پذیری:**<sup>۴</sup> ساختار شبکه عصبی به نحوی است که میتواند خود را در محیط وفق دهد به اینگونه که اگر محیط تغییر کند شبکه عصبی وزن های 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 حسگر ها باشد بروی آن تاثیر میگذارد این تطبیق پذیری باید به گونه ای باشد که نه خیلی حساس باشد که پایداری سیستم به هم بریزد نه خیلی بی تفاوت که روند یادگیری و بهبود را مختل کند.

- **تحمل خطا:**<sup>۵</sup> ساختار شبکه های عصبی به علت گستردگی به گونه ایست 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.

<sup>۲</sup> Nonlinearity

<sup>۳</sup> Input-Output Mapping

<sup>۴</sup> Adaptivity

<sup>۵</sup> Fault Tolerance

<sup>۶</sup> Neurobiological Analogy

<sup>۷</sup> Ramon y Cajal

## ۲-۴- انواع توابع فعال سازی

- **تابع Threshold:** این تابع به نوعی بر اساس اینکه مقدار ورودی میزان مشخصی را رد کرده است یا خیر تصمیم گیری انجام می دهد و بسیار پر کاربرد است:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

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

$$\varphi(v) = \begin{cases} 1 & \text{if } v > \frac{1}{4} \\ v + \frac{1}{4} & \text{if } -\frac{1}{4} \leq v \leq \frac{1}{4} \\ 0 & \text{if } v < -\frac{1}{4} \end{cases}$$

- **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 می شود.

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

- **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:

$$P(v) = \frac{1}{1 + \exp(-av)}$$

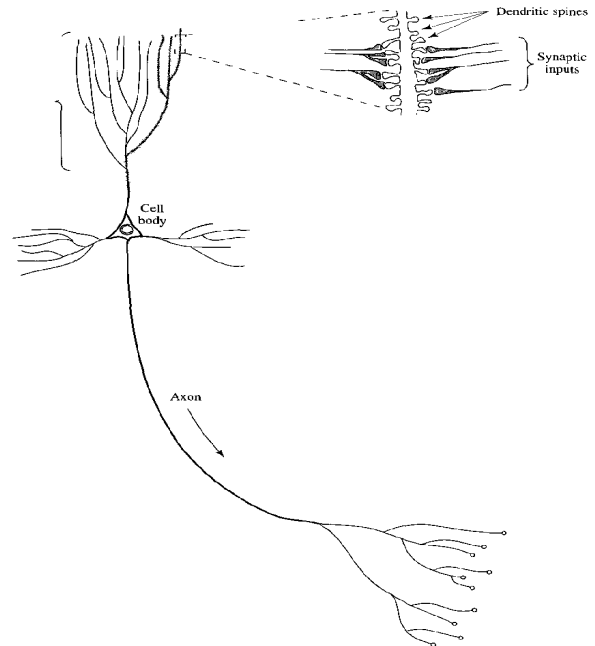
$$\varphi(v) = \begin{cases} +1 & \text{with probability } P(v) \\ -1 & \text{with probability } 1 - P(v) \end{cases}$$

## ۲-۵- نمایش شبکه عصبی با گراف جهتدار

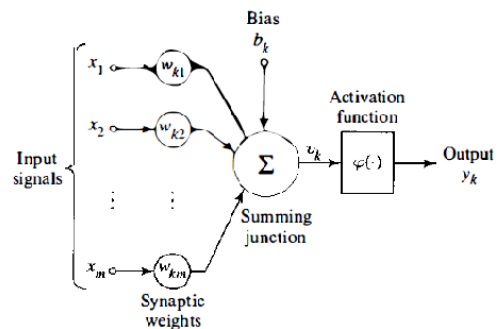
شبکه عصبی را میتوان به وسیله یک گراف جهتدار نشان داد. این گراف از سه قاعده زیر طبیعت میکند:

- **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



شکل ۲: ساختار نورون زیستی



شکل ۳: مدل ریاضی ارائه شده برای نورون

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

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

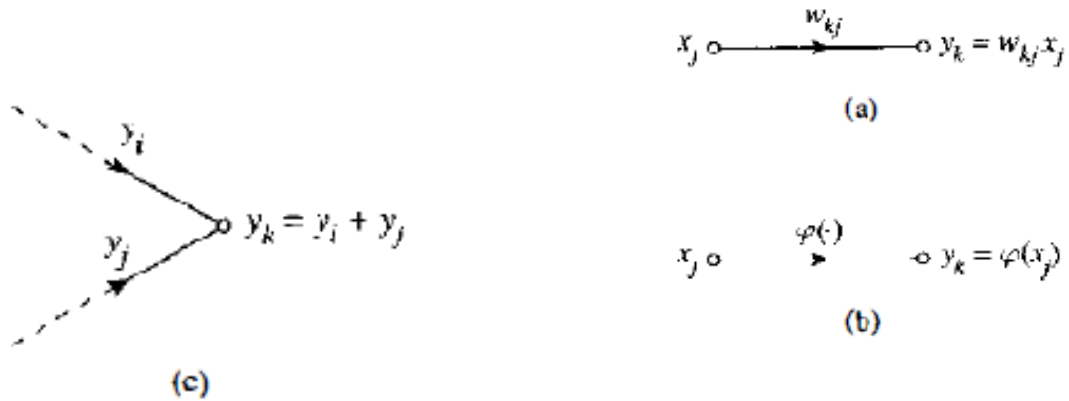
همانطور که در بالا توضیح دادیم داریم:

$$u_k = \sum_{j=1}^k W_{k,j} X_j$$

$$v_k = u_k + b_k$$

$$y_k = \varphi(v_k)$$

به مقدار  $v_k$  در عبارت بالا پتانسیل فعال سازی نورون می گوئیم.



شکل ۴: قاعده ۱

واقع همان وزن سیناپسی است ضرب میشود تا حاصل تولید شود.

– **یال فعال سازی:** 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 یال هایی است که به آن وارد میشوند.

● **قاعده ۳:** یال هایی که از یک گره خارج میشوند از هم مستقل بوده و سیگنال اولیه برابر سیگنال گره دارند.

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

شبکه عصبی یک گراف جهتدار است تشکیل شده از گره و یال هی سیناپسی یا فعال سازی و دارای چهار ویژگی زیر است:

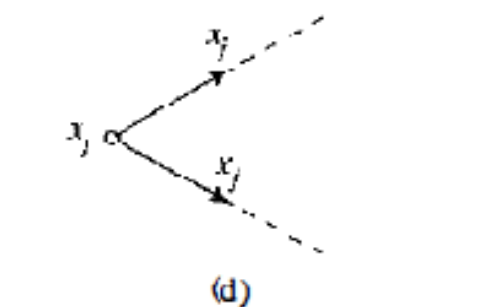
● هر نورون متشکل از تعدادی یال سیناپسی و حداکثر یک یال فعال سازی است.

● وزن یال های سیناپسی نشان دهنده وزن های سیناپسی است (مقادیر قابل تغییر در یادگیری).

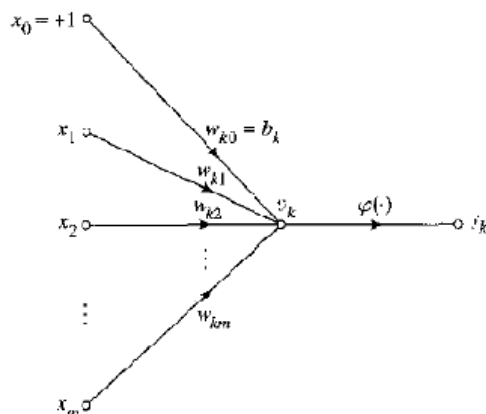
● جمع سیگنال یال های سیناپسی پتانسیل فعال سازی گره را میسازد.

● تابع فعال سازی روی پتانسیل فعال سازی اعمال میشود و خروجی نورون تولید میشود.

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



شکل ۵: قاعده ۲ و ۳



شکل ۶: مدل گراف جهتدار برای نورون

## ۲-۶- معماری شبکه عصبی

به نحوه قرار گرفتن گره در شبکه عصبی و اتصالات بین آن ها معماری شبکه عصبی می گویند. معماری شبکه های عصبی را به چند دسته ای اصلی تقسیم می کنیم:

۱. شبکه های رو به جلو: شبکه هایی که در ساختار گراف آن ها دور وجود ندارد.

- شبکه های تک لایه: شبکه های 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: شبکه هایی که حداقل دارای یک سطح بیشتر از شبکه های تک لایه هستند. ( به لایه های میانی لایه های پنهان نورون نیز میگویند)

۲. شبکه های درجریان: شبکه هایی که در ساختار گراف آن ها دور وجود دارد.

## ۳- الگوهای اتصالات

در بخش قبل، دسته بندی انواع معماری های مختلف در شبکه های عصبی انجام شد. در این بخش می خواهیم این دسته بندی ها را به تفصیل بررسی کنیم.

در بین معماری های مختلف، ساده ترین شان شبکه های تک لایه ی رو به جلو و پر قدرت ترین شان شبکه های در جریان اند. اما به دلیل آن که برای این نوع شبکه ها به صورت کلی هنوز الگوریتم های یادگیری کارآمدی طراحی نشده، از آن ها کمتر استفاده می شود و در نتیجه پرکاربردترین معماری، همان شبکه های چند لایه ی رو به جلو اند که هم توانایی بسیار بالایی دارند و هم توانایی یادگیری بالا.

البته انواع مختلفی از شبکه های در جریان خاص ساخته شده اند که الگوریتم های یادگیری خوبی نیز دارند و در موضوعاتی که مورد استفاده قرار گرفته اند قدرت بالایی از خود نشان داده اند. اما هنوز اگر الگوریتم مناسبی برای حالت کلی شبکه های در جریان پیدا شود، می تواند باعث پیشرفت بزرگی بشود.

در واقع همانطور که در بخش ۱-۲-۳- نشان داده می شود، می توان با استفاده از نورون های بسیار ساده، تمام گیت های پایه ای یک رایانه را پیاده سازی کرد، پس قدرت محاسبه ای شبکه های عصبی مصنوعی حداقل به اندازه ی قدرت رایانه ها است (حداقل در تئوری) با این تفاوت که قدرت اصلی این شبکه ها در قابلیت یادگیری آن ها بدون کمک برنامه ریزی از قبل و دانستن مسئله های پیش رو است.

شبکه های عصبی را شاید بتوان همانند با سلول های بنیادی در نظر گرفت که می توانند بدون این که برای کاری اختصاصی شده باشند، آن ها را در محیط قرار دهیم و خود به خود، خودشان را با محیط تطبیق دهند و کار را به خوبی انجام دهند.

## ۳-۱- شبکه های تغذیه ی رو به جلو

ساده ترین نوع شبکه های عصبی، شبکه هایی 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 (گره های خروجی) می رسند. معمولاً این شبکه ها را در حالت کلی به صورت شکل ۸ نشان می دهند.

اهمیت این معماری ساده در این است که با وجود سادگی، توانایی بالایی دارند و مهم تر از آن، برای آن ها الگوریتم های یادگیری کارآمد وجود دارد. در مقابل، معماری های پیچیده تری که جلوتر خواهیم دید، با وجود پتانسیل بالا نمی توانند از قدرت خود به حد کافی استفاده کنند (به جز در مواردی که برای یک دسته سوال اختصاصی شده اند).

## ۳-۲- شبکه های تک لایه

این نوع از شبکه های رو به جلو همانطور که از اسم شان مشخص است، فقط یک لایه نورون دارند (البته اگر گره های ورودی را نورون فرض نکنیم) و از آن ها با نام پرسپترون<sup>۸</sup> یاد می شود. این شبکه ها در سال ۱۹۶۰ توسط فرانک روزنبلت<sup>۹</sup> در دنباله ای از مقالات و یک کتاب<sup>۱۰</sup> بررسی شده و توسعه داده شدند. پس از انتشار این کتاب، شبکه های تک لایه محبوبیت زیادی به دست آوردند و نویسنده محبوبیت جهانی به دست آورد. پس از آن تحقیقات زیادی بر روی این شبکه ها انجام شد و محققان اکثراً انتظارات بسیار زیادی از آن ها داشتند تا این که مینسکی<sup>۱۱</sup> و پیرت<sup>۱۲</sup> در کتاب شان با نام پرسپترون<sup>۱۳</sup> در سال ۱۹۶۹ با نشان دادن محدودیت های این نوع شبکه ها، از شور و هیجان این تحقیقات به طور قابل ملاحظه ای کاستند تا حدی که تا سال ۱۹۸۰ تقریباً تحقیق بر روی این شبکه ها متوقف شد.<sup>[۷]</sup>

یک مثال قابل توجه از بزرگنمایی هایی که در مورد توانایی پرسپترون ها قبل از انتشار مقاله ی مینسکی و پیرت انجام گرفت، پروژه ای بود که در آن تعدادی عکس که در هر کدام دقیقاً یک

<sup>۸</sup>perceptron

<sup>۹</sup>Frank Rosenblatt

<sup>۱۰</sup>Principles of Neurodynamics: Perceptrons and the Theory of Brain

Mechanisms, Spartan Books, 1962

<sup>۱۱</sup>Marvin Minsky

<sup>۱۲</sup>Seymour Paper

<sup>۱۳</sup>Perceptrons: an introduction to computational geometry, 1969

تراکتور یا یک تانک (در محیط‌های مختلف) قرار داشت به یک شبکه‌ی عصبی داده شد و هدف این بود که این شبکه بتواند در 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 نوروں‌ها به سادگی قابل انجام است. اینگونه اتفاقات می‌تواند یک الگوریتم را به سرعت بدنام کنند. [۴]

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 نوروں‌ها 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 نوروں به سادگی قابل پیاده‌سازی است.

یک نکته‌ی مهم در تحلیل و بررسی رفتار شبکه‌های تک لایه این است که اگر دقت کنید، هر کدام از نوروں‌ها کاملاً به طور مستقل از دیگر نوروں‌ها رفتار می‌کند چون مقادیر ورودی را که نمی‌توانیم تغییری بدهیم و بین نوروں‌های سطح اول هم هیچ ارتباطی وجود ندارد. پس برای مثال هر شبکه‌ی تک لایه با  $k$  نوروں را می‌توان  $k$  تا شبکه با فقط یک نوروں 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.

### ۳-۲-۱- مثال‌هایی از قدرت تک نوروں

برای نشان دادن حداقل توانایی‌های یک نوروں 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 نوروں‌های این بخش و بخش بعد از نوع تابع Threshold (که در بخش ۲-۴- 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 بگوییم 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  $\frac{n}{2}$  - است.

● **فلیپ‌فلاپ<sup>۱۴</sup>:** ساخت این قطعه به داشتن دور در گراف ساختار نوروں‌ها دارد و در نتیجه فقط در شبکه‌های در جریان (بخش ۳-۴-) قابل پیاده‌سازی است. اما برای ساخت آن فقط کافی است که خروجی نوروں 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 نوروں ثابت می‌ماند و با آمدن سیگنال هم مقدار آن تعیین می‌شود.

### ۳-۲-۲- ناتوانی‌های تک نوروں

برای این که بگوییم یک نوروں‌ها 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 Threshold کار می‌کنیم)

فرض کنید نوروںی 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 نوروں را نشان می‌دهد. حال کاری که نوروں 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 (bias)) 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. [۴]

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

<sup>۱۴</sup> flip-flop



متن باز وکا<sup>۱۷</sup> مراجعه کنید.

در معماری این شبکه‌ها تعدادی لایه نرون وجود دارد که نه گره ورودی‌اند و نه گره خروجی. به این لایه‌ها در اصطلاح، لایه‌های پنهان<sup>۱۸</sup> 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.

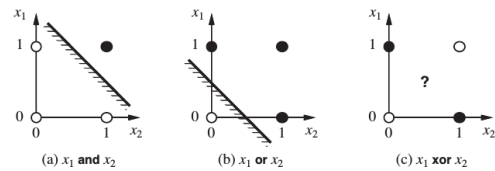
### ۳-۳-۱ - بیش‌برازش

بیش‌برازش<sup>۱۹</sup> 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.<sup>[۸]</sup> برای مثال وقتی که مقدار یک چند جمله‌ای درجه‌ی دو را با یک چند جمله‌ای درجه‌ی سه تخمین بزنیم.

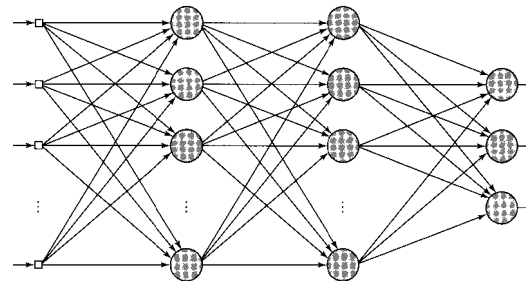
همانند تمام مدل‌های آماری، شبکه‌های چند لایه نیز وقتی که تعداد پارامترهای مسئله (همان تعداد لایه‌ها و اندازه‌ی کلی شبکه) بیش از حد باشد، به مشکل بیش‌برازش بر می‌خوریم. در این حالت حتی شبکه‌ی ما می‌تواند به صورتی تمام ورودی‌های داده‌شده را در خود ذخیره کند و بر روی آن‌ها درست جواب بدهد اما لزوماً این نتایج را بر روی ورودی‌های جدید تعمیم ندهد. به وضوح اگر اندازه‌ی شبکه بسیار کوچک باشد نیز مشکلات دیگری<sup>۲۰</sup> 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<sup>۲۱</sup> است. یک روش برای شبکه‌های تماماً متصل نیستند، الگوریتم صدمه‌ی مغزی بهینه<sup>۲۲</sup> است که ابتدا از یک گراف کامل شروع می‌کند و سپس بعضی از ارتباطات آن‌را حذف می‌کند.<sup>[۲]</sup>

ص-۷۳۷]



شکل ۷: جدایی ناپذیری خطی XOR (کتاب راسل<sup>[۲]</sup>، ص-۷۳۰)



شکل ۸: معماری شبکه‌های چند لایه (این مثال: سه لایه)

نقاط آن (مثلاً فقط نقاطی که خروجی‌شان یک است) خاصیت جدایی‌پذیری خطی<sup>۱۵</sup> داشته باشند.

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

### ۳-۳-۳ - 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 تصمیم‌گیری 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<sup>۱۶</sup> 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

<sup>۱۷</sup><http://www.cs.waikato.ac.nz/ml/weka/>

<sup>۱۸</sup>hidden layers

<sup>۱۹</sup>overfitting

<sup>۲۰</sup>underfitting

<sup>۲۱</sup>cross-validation

<sup>۲۲</sup>optimal brain damage

<sup>۱۵</sup>Linear separability

<sup>۱۶</sup>Data Mining

این نوع شبکه‌ها بسیار قدرتمندتر از شبکه‌های رو به جلو اند. در کل در طراحی شبکه‌های در جریان<sup>۲۶</sup> می‌تواند دوره‌های 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 ۳-۲-۱- نشان داده شد) را دارند. و همان طور که در بخش مذکور نشان داده شد، می‌تواند تمام اجزای پایه‌ای یک رایانه‌ی کامل را داشته باشند. پس اگر بتوانیم آن‌ها را تعلیم دهیم قدرت بسیار بالایی می‌تواند داشته باشند. اما به دلیل این که می‌تواند 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. معمولاً 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 با این کار می‌توانیم از الگوریتم‌های یادگیری ساخته شده برای شبکه‌های رو به جلو، در این شبکه‌ها نیز استفاده کنیم.<sup>[۴]</sup>

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

### ۳-۵- 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<sup>۲۷</sup> 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.<sup>[۴]</sup>

یکی از ابتدایی‌ترین 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<sup>۲۳</sup> 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 نوروها توابعی پیوسته باشند. برای مثال می‌توان از تابع فعال‌سازی Sigmoid (بخش ۲-۴-) استفاده کرد.

برای بهتر فهمیدن روش این الگوریتم، در نظر بگیرید که خروجی نوروهای لایه‌های جلوتر یک تابع بر حسب خروجی نوروهای 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.<sup>[۴]</sup>

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<sup>۲۴</sup> 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<sup>۲۵</sup> 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 معمولاً ابتدا برای لایه‌های اولیه از الگوریتم‌های یادگیری بدون ناظر استفاده می‌شود و سپس برای لایه‌های انتهایی از این الگوریتم یا الگوریتم‌های مشابه استفاده می‌شود.

<sup>۲۳</sup> back propagation

<sup>۲۴</sup> deep networks

<sup>۲۵</sup> Vanishing Gradient problem

<sup>۲۶</sup> recurrent

<sup>۲۷</sup> Ilya Sutskever



### ۳-۵-۱- 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<sup>۲۸</sup> 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<sup>۲۹</sup> در بخش ۳-۲-۵) بر اساس همین موضوع شبکه‌هایی را طراحی کردند. شبکه‌ی هاپفیلد معمولاً به عنوان حافظه‌های تداعی‌گیر<sup>۳۰</sup> استفاده می‌شوند. ثابت می‌شود که این شبکه‌ها همیشه به یک کمینه‌ی موضعی<sup>۳۱</sup> 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.<sup>[۹]</sup>

مسئله‌ی حافظه‌ی تداعی‌گر را می‌توان به نوعی ساده‌ترین مسئله برای نشان دادن 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.<sup>[۶، ص-۱۱ و ۱۲]</sup>

### ۳-۵-۲- ماشین بلتزمن

هینتون<sup>۳۲</sup> و سجنوفسکی<sup>۳۳</sup> در سال ۱۹۸۵ یک قانون یادگیری جامع برای شبکه‌های احتمالاتی متقارن ابداع کرده (بخش ۵-۳- ) 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<sup>۳۴</sup> سرعت یادگیری این ماشین را به طور چشم‌گیری افزایش می‌دهد.<sup>[۶، ص-۱۶۳]</sup> ماشین بلتزمن معمولاً 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.

### ۴- تعریف مسئله

جستجو و یافتن الگو درون داده‌ها را می‌توان یک مسئله‌ی کاملاً پایه‌ای و پر کاربرد در علم، صنعت و به طور کلی در زندگی در نظر گرفت. برای مثال در قرن ۱۶ ام کیپلر با توجه به اطلاعات زیاد مشاهداتی موجود متوجه الگوی حرکتی سیارات شد و با مشاهدات طیف نشی اتم‌ها به پیدایش فیزیک کوانتوم ختم شد.

به طور کلی فیلد شناسایی الگوها به دنبال شناسایی خودکار نظم و قواعد درون داده‌ها توسط الگوریتم‌های کامپیوتری و پس از پیدا کردن این قواعد، به دنبال کارهایی همچون دسته‌بندی این داده‌ها در گروه‌های مختلف است.<sup>[۳، ص-۱]</sup>

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

معیارهایی که با آن‌ها می‌توان این مسائل را تفکیک کرد متنوع اند، اما از میان آن‌ها می‌توان رایج‌ترین و مهم‌ترین آن‌ها را نام برد که عبارت‌اند از، **نحوه‌ی یادگیری** که معمولاً این مسائل را در دو فرم با نظارت و بدون نظارت دسته‌بندی می‌کند و **خواسته‌ی مسئله** که به طور عمده یکی از حالت‌های دسته‌بندی داده‌ها و یا پیش‌بینی داده‌های پیوسته است.

در ادامه هر یک از این معیارها را بررسی می‌کنیم.

### ۴-۱- خواسته‌ی مسئله

الگویی که بر روی یک دسته از داده‌ها پیدا می‌شود را می‌توان یافته‌ی اصلی الگوریتم در نظر گرفت و با توجه به انواع الگوهایی که یک الگوریتم شناسایی می‌کند، می‌توان الگوریتم‌ها و مسائل متناظر آن‌ها را دسته‌بندی کرد.

این الگو در واقع یک تابع است که الگوریتم با بررسی داده‌های قدیمی پیدا می‌کند و از این تابع برای حدس زدن ویژگی‌های

<sup>۲۸</sup>John Hopfield

<sup>۲۹</sup>Boltzmann

<sup>۳۰</sup>Content-addressable memory

<sup>۳۱</sup>local minima

<sup>۳۲</sup>Hinton

<sup>۳۳</sup>Sejnowski

<sup>۳۴</sup>mean field

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

به طور کلی در یک مسئله یادگیری<sup>۳۵</sup> در صورتی که خواسته‌ی مسئله درون یابی<sup>۳۶</sup> یا برون یابی<sup>۳۷</sup> یک داده‌ی پیوسته باشد به آن یک مسئله رگرسیون<sup>۳۸</sup> می‌گویند و در صورتی که هدف پیشینی یک کلاس گسسته و محدود برای داده‌های جدید باشد به آن کلاس‌بندی<sup>۳۹</sup> می‌گویند، یا به تفسیری دیگر از کتاب پترن ریکانیشن:

در صورتی که هدف یک فرایند، انتساب داده‌های 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 رگرسیون می‌گوییم.<sup>[۳]</sup>

در نتیجه این دسته‌بندی بر اساس خروجی تابع  $y$  است.

## ۲-۴- نحوه‌ی یادگیری

فرض کنید یک مجموعه‌ی داده  $X = x_1, x_2, \dots, x_N$  در اختیار شما قرار گرفته و از شما می‌خواهند که با استفاده از آن یک روش و یا الگو ارائه دهید که در صورت مواجهه با داده‌ی جدید  $\hat{x}$  با آن الگو بتوان یک ویژگی از  $\hat{x}$  به اسم  $\hat{t}$  را با تابع  $y(\hat{x})$  پیشینی 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<sup>۴۰</sup> نام دارد یک بردار  $t = t_1, t_2, \dots, t_N$  به نام بردار هدف<sup>۴۱</sup> از مقدار ویژگی مورد نظربه ازای ورودی‌های متناظر سوال به ما داده می‌شود، و هدف آن است که با یافتن یک الگو در این ۲ مجموعه  $y(x)$  را شکل دهیم و این ویژگی را برای ورودی‌هایی که  $\hat{t}$  به ما داده نشده با  $y(\hat{x})$  پیشینی کنیم، به یادگیری این دسته از مسائل به علت راهنمایی‌هایی که در مرحله‌ی یادگیری به الگوریتم داده می‌شود، یادگیری با نظارت<sup>۴۲</sup> 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 ایده‌ای از خواسته‌ی مسئله به دنبال ویژگی‌ای در مسئله بگردد، اما در ادامه می‌بینیم در واقعیت 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 تفاوت‌هایی مانند شکل، اندازه و ... آن‌ها را در گروه‌هایی دسته‌بندی می‌کند و برای هر گروه مفهومی در ذهن خود در نظر 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.

## ۵- یادگیری

بر اهمیت ترین ویژگی شبکه‌های 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 عصب‌ها انجام می‌شود. تعریف دقیق یادگیری در شبکه‌های عصبی را می‌توان 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.<sup>[۱، ص-۵۰]</sup>

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 صورت‌های متفاوتی می‌توانند شکل گیرند، می‌توان انواع زیادی از یادگیری را تولید کرد، چند نمونه از روش‌های یادگیری معروف را می‌توان روش‌های زیر دانست:

۱. تصحیح خطا
۲. یادگیری هب
۳. یادگیری رقابتی
۴. یادگیری بر مبنای حافظه
۵. یادگیری بتلزن

<sup>۳۵</sup> Learning Problem

<sup>۳۶</sup> Interpolation

<sup>۳۷</sup> Extrapolation

<sup>۳۸</sup> Regression

<sup>۳۹</sup> Classification

<sup>۴۰</sup> Training set

<sup>۴۱</sup> Target vector

<sup>۴۲</sup> Supervised learning

The diagram shows a central node labeled  $d$  enclosed in a dashed circle. Inside this circle are several nodes labeled '1'. Outside the circle are several nodes labeled '0'. An arrow points from the word 'outlier' to the central node  $d$ .

در ساده ترین حالت این روش یادگیری می توانید فرض کنید یک عصب  $k$  داریم که خروجی آن قابل اندازه گیری است و آن را با  $y(x)$  نشان می دهیم، برای این عصب یک مقدار مطلوب خروجی وجود دارد که آن را  $d(x)$  می نامیم و مستقیماً به ما داده می شود. در اثر مقایسه ی سیگنال خروجی با سیگنال مطلوب یک سیگنال خطا به دست می آید که برابر با مقدار زیر است:

۵-۲- یادگیری بر مبنای حافظه

$$\xi(n) = \frac{1}{\gamma} e_k^{\gamma}(n)$$
$$\Delta\omega_{k,i} = \eta e_k(n)x_i(n)$$

$$\min_i d(x_i, \hat{x}) = d(x'_N, \hat{x})$$

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

<sup>43</sup>Nearest neighbor rule

در این روش همسایگی  $\hat{x}$ ،  $k$  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.

### ۳-۵ 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 ۳-۵-۲) می‌گویند.

در یک ماشین بلتزمن هر عصب می‌تواند در ۲ حالت مختلف مثلاً ۱ یا ۰ 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:

$$E = -\frac{1}{2} \sum_j \sum_{k \neq j} \omega_{kj} x_k x_j$$

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).

$$P(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E_k/T)}$$

اینجا  $\Delta E_k$  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 می‌کند:

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

۲. حالتی که عصب های آشکار آزادانه تغییر می‌کنند.

که در هر دو این حالات وضعیت عصب های پنهان به صورت آزاد می‌تواند تغییر کند. فرض کنید مقدار هم بستگی وضعیت های دو عصب  $i$  و  $j$  و میانگین گرفته شده در تمام وضعیت های 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.

### ۴-۵ هبین (Hebians)

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، ص-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، ص-۵۵]

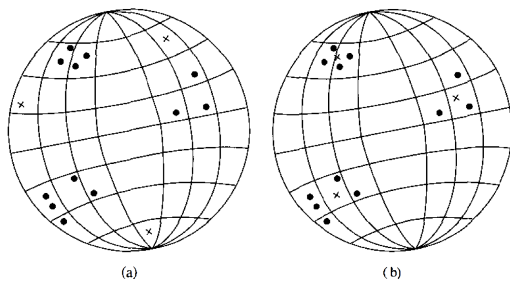
به چنین سیناپسی سیناپس هبین می‌گویند، در واقع چنین اتصالی از یک فرایند فعال وابسته به زمان و به شدت موضعی برای بهبود کارایی عصب ها و بهینه سازی آنها استفاده می‌کند.

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 کواریانس 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.

### ۵-۵ 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



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، ص-۶۰])

شکل ۱۰ این فرایند را نشان می دهد.

## ۶- کاربردهای مهم

در چند دهه ی اخیر بسیاری از فعالیت هایی که پیش از این تنها به وسیله ی انسان قابل انجام بوده، به وسیله ی کامپیوتر دست یافتنی شده است. علت این پیشرفت را می توان پیدایش پردازنده های قوی تر و همچنین الگوریتم های جدید ماشین لرنینگ دانست. به عنوان یکی از این الگوریتم ها شبکه های عصبی نیز نقش موثری در این فرایند داشته اند. مواردی که در ادامه مطرح می شود به طور کلی کاربرد هایی هستند که به وسیله ی شبکه های عصبی و یا سایر ماشین های یادگیری آماری ممکن شده اند.

### ۶-۱- 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 شبکه های عصبی بود ولی با پیدایش شبکه های عمیق و پیشرفت شبکه های عصبی، این شبکه ها از ماشین های آماری پیشی گرفتند.

در این مورد معرفی چند مورد مناسب به نظر می رسد:

۱. در الگوریتم های داده کاوی برای پیدا کردن الگوها و گشتن به دنبال بی قاعدگی ها به طور گسترده از شبکه های چند لایه (بخش ۳-۳) استفاده می شود.

۲. برای پیدا کردن الگو مشابه در یک بانک از الگوها می توان از شبکه های هاپفیلد (بخش ۳-۵-۱) استفاده کرد.

۳. برای بازسازی عکس های آسیب دیده یا ناقص و یا بازیابی

دسته بندی داده ها مناسب می سازد. قواعد این روش را در سه عنصر می توان خلاصه کرد:

۱. یک مجموعه از عصب ها که کاملاً 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 وسیله ی آن بتوانند رقابت کنند و برنده تعیین شود.

در ساده ترین حالت این شبکه می توان یک سطح از عصب های خروجی در نظر گرفت که همه ی عصب های ورودی به همه ی آن ها اتصالاتی با وزن مثبت دارند (اتصالات بین عصب های خروجی نیز مشکلی ایجاد نمی کند).

برای این که عصب  $k$  بین عصب ها برنده شود باید میدان القایی<sup>۴۴</sup> آن به ازای ورودی  $\hat{x}$  در بین همه ی عصب ها بیشینه باشد، در نتیجه ورودی های عصب برنده به نحوی تغییر می کنند که عصب  $k$  با ورودی  $\hat{x}$  آموزش یابد و این عصب با این ورودی راحت تر تحریک شود، برای این کار بردار وزن اتصالات عصب  $k$  به سمت ورودی  $\hat{x}$  شیفت داده می شود، یا به عبارتی:

$$\Delta\omega_{k,j} = \begin{cases} \eta(x_j - \omega_{k,j}) & \text{اگر عصب } k \text{ مسابقه را ببرد} \\ 0 & \text{اگر عصب } k \text{ مسابقه را ببازد} \end{cases}$$

که ضریب  $\eta$  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 وسیله ی شبکه ی عصبی بالا در گروه های 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 می کنیم و در صورتی که بردار ورودی یک عصب به ورودی یک آزمایش نزدیک تر باشد (و در نتیجه بیشتر از بقیه ی عصب ها تحریک شود)، آن عصب را متناظر با گروه آن ورودی در نظر می گیریم و بردار آن را به بردار این ورودی نزدیک می کنیم. بعد از چندین مرحله اجرای این آزمایش به حالتی می رسیم که بردار هر عصب در میان گروه متناظر آن قرار دارد و در مراحل بعدی این عصب اعضای آن گروه را شناسایی می کند، 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 برنده ی خود را خواهد داشت و بردار عصب های دیگر را به سمت خود نمی کشاند.

<sup>۴۴</sup>Local induced field

۱. تقریب تابع‌ها: تقریب زنده‌توابعی که محاسبه‌ی آن‌ها به راحتی امکان پذیر نیست.

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

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

## مراجع

- [1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2<sup>nd</sup> ed., New Jersey, Prentice-Hall, 1999.
- [2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3<sup>rd</sup> ed., New Jersey, Prentice-Hall, 2010
- [3] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1<sup>st</sup> ed., New York, Springer, 2006.
- [4] G. Hinton, *Neural Networks and Machine Learning [lecture notes]*, Retrieved from coursera.org, 2012
- [5] A. Ng, *Machine Learning [lecture notes]*. Retrieved from coursera.org, 2014
- [6] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the theory of Neural Computation*, Lecture notes Volume I, California, Addison-Wesley publishing company, 1990.
- [7] Wikipedia contributors, "Frank Rosenblatt" *Wikipedia, The Free Encyclopedia*, [online], Available: [https://en.wikipedia.org/wiki/Frank\\_Rosenblatt](https://en.wikipedia.org/wiki/Frank_Rosenblatt)
- [8] Wikipedia contributors, "Overfitting" *Wikipedia, The Free Encyclopedia*, [online], Available: <https://en.wikipedia.org/wiki/Overfitting>
- [9] Wikipedia contributors, "Hopfield network" *Wikipedia, The Free Encyclopedia*, [online], Available: [https://en.wikipedia.org/wiki/Hopfield\\_network](https://en.wikipedia.org/wiki/Hopfield_network)

اطلاعات متون خطی که تا حدی از دست رفته‌اند می‌توانیم از شبکه‌های بلترمن (بخش ۳-۵-۲-) یاری بجویم.

## ۲-۶- خوشه‌سازی و دسته‌بندی

وقتی حجم داده‌ها زیاد و یا تعداد ابعاد ورودی بیش از 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 مختلف می‌توانند تا حد خوبی خصوصیات و ویژگی‌های مختلف داده‌ها را از یکدیگر تشخیص بدهند.

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

## ۱-۲-۶- استخراج اطلاعات

انسان‌ها در برابر حجم زیاد داده‌ها از ماشین‌ها عقب می‌مانند، علاوه بر این، گاهی ماشین‌ها نظمی در داده‌ها 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.

## ۲-۲-۶- تحلیل اطلاعات

کامپیوترها در این زمینه هم ثابت کردند که در بعضی از موارد از انسان‌ها سریع‌ترند، بسیاری از شرکت‌های اقتصادی بر پایه‌ی این برنامه‌ها سرمایه‌گذاری‌های خود را انجام می‌دهند.

## ۳-۶- بهینه‌سازی مسائل پیچیده

مسائل NP دسته‌ای کاربردی از مسائل هستند که تا کنون الگوریتم‌ای قطعی برای حل آن‌ها در زمان مناسب ارائه نشده است. اما شبکه‌های عصبی و سایر ماشین‌های یادگیری می‌توانند آن‌ها را در زمان خیلی خوبی با اطمینان بالایی حل کنند. و شاید حتی این دسته از مسئله‌ها را بتوان علت این دانست که چرا الگوریتم‌های قطعی توانایی رقابت با مغز انسان را ندارند.

## ۴-۶- سایر کاربردها

چند نمونه از دیگر موارد کاربردی شبکه‌های عصبی:

<sup>۴۵</sup>Long Short-term Memory