

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 -

کلید واژه‌ها - الگوریتم، هوش مصنوعی، شبکه‌های عصبی.

۱ - مقدمه

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 دارا می‌باشد.

برای مثال قدرت بینایی و بصری انسان را در نظر بگیرید که یک نوع پردازش اطلاعات است. ما اطلاعات رواز محیط بیرون توسط حسگر پیچیده چشم دریافت میکنیم آن هارا 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 میلی ثانیه 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 انعکاس 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 اندازه ی یک آلو است انجام می پذیرد.

اما مغز انسان یا خفاش چگونه این کار هارا انجام می دهد ؟

۲ - آشنایی و مفاهیم اولیه

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

شبکه عصبی مصنوعی یک پردازنده توزیع شده موازی و گسترده است و که از واحد های پردازشی ساده ساخته شده است

که میتواند دانش اکتسابی را در خود ذخیره کند و در آینده از آن ها در تصمیم گیری ها 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 میگوئیم برای ذخیره اطلاعات استفاده میشود.

۱-۲ مزایای شبکه های عصبی

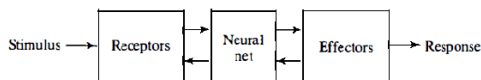
شبکه عصبی قدرت محاسباتی اش را از دو ویژگی بهره میگیرد ویژگی اول اینکه یک شبکه بسیار گسترده، موازی و توزیع شده است و ویژگی دوم اینکه قابلیت یادگیری است اینکه میتواند با توجه به محیط یک سری اطلاعات و تجارب کسب کند و از این تجارب در تصمیم گیری های بعدی استفاده کند به این ویژگی کلیت بخشی^۱ میگوئیم. این ویژگی ها باعث میشود مسائلی که در حال حاضر برای کامپیوتر های مرسوم دست نیافتی است توسط این شبکه ها قابل حل باشد البته باید توجه داشت که گونه های خاصی از مسائل هستند که توسط این شبکه های عصبی به طور بهینه قابل حل هستند و پردازنده های مرسوم در بعضی موارد بهتر 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:

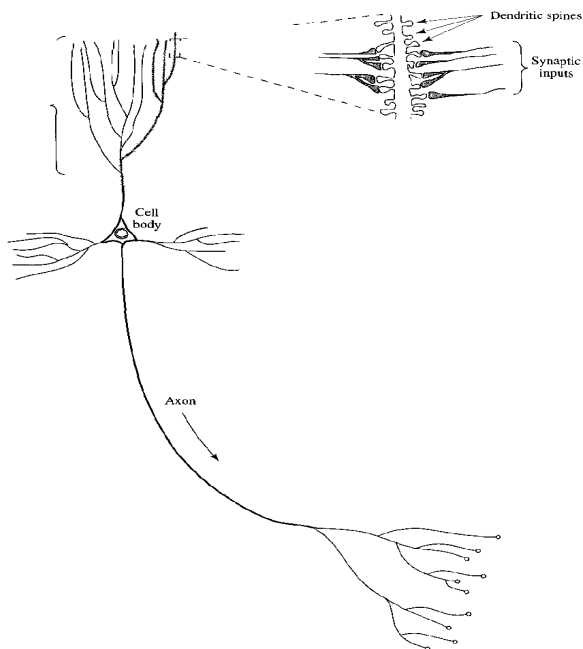
- توانایی غیرخطی بودن:^۲ واحد های سازنده شبکه عصبی یا همان نورون ها میتوانند غیرخطی باشند و در نتیجه کل

^۱ Generalization

^۲ Nonlinearity



شکل ۱: ساختار سه بخشی شبکه عصبی انسان



شکل ۲: ساختار نورون زیستی

بخش مرکزی همان شبکه عصبی است که شامل شبکه گسترده نورون ها است و محل تصمیم گیری می باشد. 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^۷ برای مغز معرفی شد. نورون ها ۵ الی ۶ مرتبه توانی از سیلیکون کند تر هستند. اتفاقات در چیپ های سیلیکونی هر ۱ نانوثانیه اتفاق می افتد 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 تر 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 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

^۷Ramon y Cajal

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 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:**^۴ 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:**^۵ 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.

● **آزمایشات زیستی:**^۶ 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.

^۳ Input-Output Mapping

^۴ Adaptivity

^۵ Fault Tolerance

^۶ Neurobiological Analogy

- **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 Threshold 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

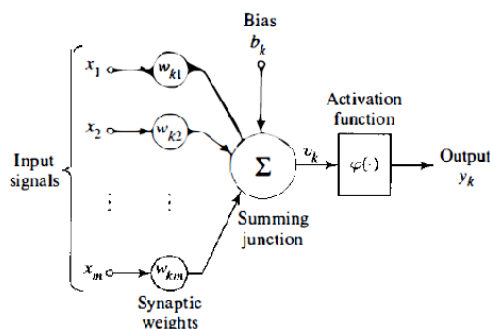
شبکه عصبی را میتوان به وسیله یک گراف جهتدار نشان داد. این گراف از سه قاعده زیر طبیعت میکند:

- **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 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.



شکل ۳: مدل ریاضی ارائه شده برای نورون

می کنید. در اینجا به تشریح بخش های مختلف می پردازیم:

- **سیناپس ها یا اتصالات** هر کدام به همراه یک عدد که به آن وزن سیناپسی میگوئیم 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.

- **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:

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

$$v_k = u_k + b_k$$

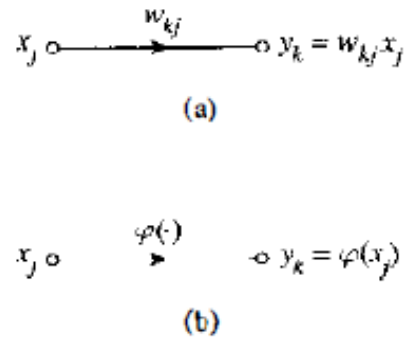
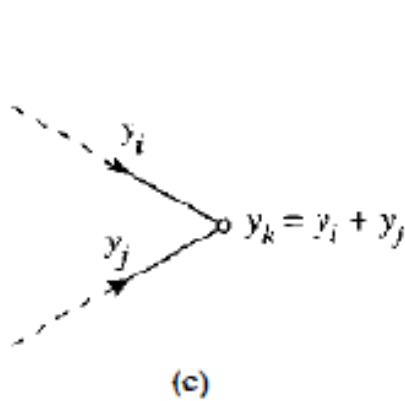
$$y_k = \varphi(v_k)$$

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

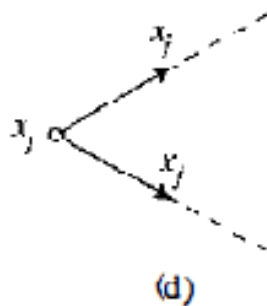
۲-۴- انواع توابع فعال سازی

- **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:

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



شکل ۴: قاعده ۱



شکل ۵: قاعده ۲ و ۳

- 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 میتوانیم 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 های سیناپسی یا فعال سازی و دارای چهار ویژگی زیر است:

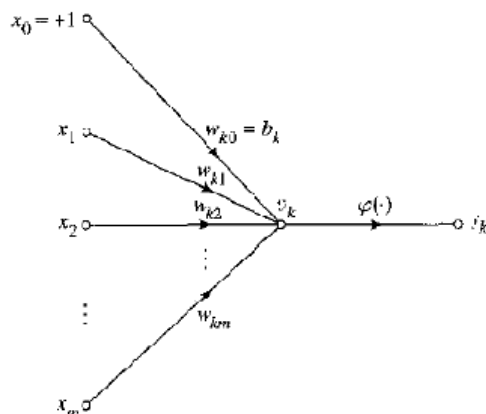
- هر نورون متشکل از تعدادی یال سیناپسی و حداکثر یک یال فعال سازی است.

- 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 میتوانیم مدل نورون ارائه شده را به عنوان یک گراف جهتدار مشاهده کنید.



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 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 ان 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 درجریان 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 درجریان 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 درجریان پیدا شود، می تواند باعث پیشرفت بزرگی بشود.

در واقع همانطور که در بخش ۳-۲-۱- نشان داده می شود، می توان با استفاده از نرون 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. معمولا 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^۸ یاد می شود. این شبکه ها در سال ۱۹۶۰ توسط فرانک روزنبلت^۹ در دنباله ای از مقالات و یک کتاب^{۱۰} 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 اکثرا انتظارات بسیار زیادی از آن ها داشتند تا این که مینسکی^{۱۱} و پیرت^{۱۲} در کتاب شان با نام پرسپترون^{۱۳} 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 تقریبا تحقیق بر روی این شبکه ها متوقف شد.^[۷]

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 دقیقا یک تراکتور یا یک تانک (در محیط های مختلف) قرار داشت به یک شبکه ی عصبی داده شد و هدف این بود که این شبکه بتواند در نهایت تشخیص بدهد که وسیله ی موجود در عکس تانک بوده یا تراکتور. حتی در بعضی از این عکس ها قسمتی از وسیله توسط درختان جنگلی پوشیده شده بود و وسیله به طور کامل دیده نمی شد. این پروژه در ابتدا نتایج بسیار موفقیت آمیزی داشت و به خوبی می توانست 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 تانک ها در یک روز

^۸perceptron

^۹Frank 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

ابری گرفته شده بود و تنها کاری که شبکه انجام می داد تعیین مقدار روشنایی تصویر و تصمیم گیری بر اساس آن بود، کاری که برای چشم انسان (حداقل با دقت بالا) کار ساده ای نیست، اما برای نورون ها به سادگی قابل انجام است. اینگونه اتفاقات می توانند یک الگوریتم را به سرعت بدنام کنند. [۴]

ساده ترین مثالی که شبکه های تک لایه در شبیه سازی شان عاجز اند، گیت منطقی xor است. البته این تنها مثال نیست و فقط نماینده ی گروه بزرگی از توابع است که نورون ها قابلیت تولید خروجی همانندشان را ندارند، اما خبر خوب این است که این مشکل با اضافه کردن تعداد لایه ها حل می شود و همین گیت xor با دو لایه نورون به سادگی قابل پیاده سازی است.

یک نکته ی مهم در تحلیل و بررسی رفتار شبکه های تک لایه این است که اگر دقت کنید، هر کدام از نورون ها کاملاً به طور مستقل از دیگر نورون ها رفتار می کند چون مقادیر ورودی را که نمی توانیم تغییری بدهیم و بین نورون های سطح اول هم هیچ ارتباطی وجود ندارد. پس برای مثال هر شبکه ی تک لایه با k نورون را می توان k تا شبکه با فقط یک نورون در نظر گرفت که البته ورودی تمام شان را یکی می دهیم. این موضوع باعث می شود که تحلیل این شبکه ها بسیار ساده باشد و الگوریتم یادگیری کارایی برای آن ها طراحی شود. البته این موضوع باعث می شود که به سادگی ببینیم که این نوع شبکه ها توانایی بالایی در حل مسائل مختلف ندارند و با بزرگتر کردن شان بر قدرت شان افزوده نمی شود.

۳-۲-۱- مثال هایی از قدرت تک نورون

برای نشان دادن حداقل توانایی های یک نورون ساده، با استفاده از آن ها گیت های منطقی پایه ای رایانه و دو مثال پیشرفته تر را طراحی می کنیم. برای این کار فرض کنید که تابع فعال سازی تمام نورون های این بخش و بخش بعد از نوع تابع Threshold (که در بخش ۲-۴ تعریف شده) می باشد:

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

● **گیت and:** برای ساختن این گیت باید دو ورودی داشته باشیم که ضریب هر کدام را برابر ۱ قرار می دهیم و مقدار عدد ثابت را برابر ۱/۵ قرار می دهیم. 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 می دهیم. حال اگر هر کدام را یک کنیم، جواب یک می شود. (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 آن ها صفر بوده اند یا یک: برای این تابع هم n ورودی داریم که ضریب هر کدام یک است و مقدار عدد ثابت برابر $\frac{n}{2}$ است.

● **فلیپ فلاپ^{۱۴}:** ساخت این قطعه به داشتن دور در گراف ساختار نورون ها دارد و در نتیجه فقط در شبکه های در جریان (بخش ۳-۴) قابل پیاده سازی 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 می تواند ۱- هم باشد) و عدد ثابت (bias) هم برابر ۱- باشد. حال تا وقتی که سیگنالی نیاید، همیشه مقدار خروجی نورون ثابت می ماند و با آمدن سیگنال هم مقدار آن تعیین می شود.

۳-۲-۲- ناتوانی های تک نورون

برای این که بگوییم یک نورون ها نمی تواند چه کارهایی را انجام دهد، ساده تر است که نشان دهیم چه کارهایی می تواند بکند و هر چه در آن قالب نگنجد، کارهایی است که نمی تواند انجام دهد. (در این بخش فقط با توابع فعال سازی Threshold کار می کنیم)

فرض کنید نورونی که می خواهیم 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 نقطه های این فضا n بعدی یک حالت از مقادیر ورودی نورون را نشان می دهد. حال کاری که نورون ما باید انجام بدهد این است که به ازای هر نقطه، مشخص کند که خروجی اش باید صفر باشد یا یک. حال با مقدار دهی به وزن ها (و مقدار ثابت (bias)) ما در واقع یک ابر صفحه ی n بعدی انتخاب می کنیم و به نقاطی که در یک طرف صفحه باشند مقدار یک و باقی نقاط مقدار صفر را اختصاص می دهیم. [۴]

با استفاده از این تعریف متوجه می شویم که یک نورون فقط در زمانی می تواند یک تابع با خروجی صفر و یک را تولید کند که نقاط آن (مثلاً فقط نقاطی که خروجی شان یک است) خاصیت جدایی پذیری خطی^{۱۵} داشته باشند.

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

۳-۳- شبکه های چند لایه

این شبکه ها که حالت کلی شبکه های رو به جلو اند، امروزه مدل استاندارد مورد استفاده برای بیشتر مسائل تشخیص الگو با

^{۱۴} flip-flop

^{۱۵} Linear separability

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

۳-۳-۱ - 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. [۸] برای مثال وقتی که مقدار یک چند جمله‌ای درجه‌ی دو را با یک چند جمله‌ای درجه‌ی سه تخمین بزنیم.

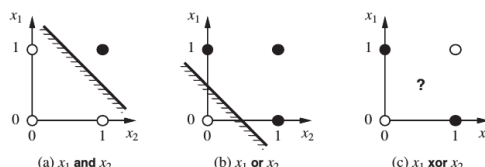
همانند تمام مدل‌های آماری، شبکه‌های چند لایه نیز وقتی که تعداد پارامترهای مسئله (همان تعداد لایه‌ها و اندازه‌ی کلی شبکه) بیش از حد باشد، به مشکل بیش‌برازش بر می‌خوریم. در این حالت حتی شبکه‌ی ما می‌تواند به صورتی تمام ورودی‌های داده‌شده را در خود ذخیره کند و بر روی آن‌ها درست جواب بدهد اما لزوماً این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

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

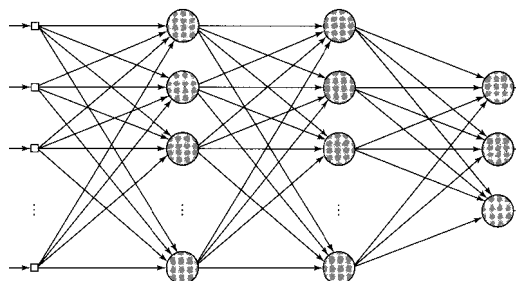
اگر قرار باشد در شبکه موجود تمام گره‌ها به یکدیگر وصل باشند، تنها موضوع باقی‌مانده، اندازه‌ی شبکه است. یکی از روش‌های ساده برای حل این مشکل این است که ابتدا چند ساختار را بررسی کنیم و کوچک‌ترین ساختاری را انتخاب کنیم که نتایج قابل قبولی دارد. روش‌های دیگر استفاده از روش اعتبارسنجی متقابل^{۲۱} است. یک روش برای شبکه‌های تماماً متصل نیستند، الگوریتم صدمه‌ی مغزی بهینه^{۲۲} است که ابتدا از یک گراف کامل شروع می‌کند و سپس بعضی از ارتباطات آن‌را حذف می‌کند. [۲، ص-۷۳۷]

۳-۳-۲ - انتشار رو به عقب

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^{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



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



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

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^{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} 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^{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.

^{۱۹}overfitting

^{۲۰}underfitting

^{۲۱}cross-validation

^{۲۲}optimal brain damage

^{۲۳}back propagation

^{۱۶}DataMining

^{۱۷}<http://www.cs.waikato.ac.nz/ml/weka/>

^{۱۸}hidden layers

می‌کند. یک پیش‌نیاز برای انجام این روش این است که توابع فعال‌سازی نورون‌ها 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 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. [۴]

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 گرایان 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 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.

۳-۴- شبکه‌های در جریان

این نوع شبکه‌ها بسیار قدرتمندتر از شبکه‌های رو به جلو اند. در کل در طراحی شبکه‌های در جریان^{۲۶} 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، 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 طبعی‌ترین 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. [۴]

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 محدودیت‌های شان، 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^{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. [۴]

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^{۲۸} و دیگران متوجه شدند که تحلیل این نوع شبکه‌ها بسیار ساده‌تر از حالت کلی شبکه‌های در جریان است. در این شبکه‌ها علاوه بر دو طرفه بودن یال‌ها، یال به خود (طوقه) هم نداریم.

هاپفیلد و دیگران (همانند ماشین بلتزمن^{۲۹} در بخش ۳-۵-۲-) بر اساس همین موضوع شبکه‌هایی را طراحی کردند. شبکه‌ی هاپفیلد معمولاً به عنوان حافظه‌های تداعی‌گیر^{۳۰} استفاده

^{۲۷} Ilya Sutskever

^{۲۸} John Hopfield

^{۲۹} Boltzmann

^{۳۰} Content-addressable memory

^{۲۴} deep networks

^{۲۵} Vanishing Gradient problem

^{۲۶} recurrent

۴- تعریف مسئله

جستجو و یافتن الگو درون داده‌ها را می‌توان یک مسئله‌ی کاملاً پایه‌ای و پرکاربرد در علم، صنعت و به‌طور 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 طیف‌نشری اتم‌ها به پیدایش فیزیک کوانتوم ختم شد.

به‌طور 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. [۳، ص-۱]

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

معیارهایی که با آن‌ها می‌توان این مسائل را تفکیک کرد متنوع‌اند، اما از میان آن‌ها می‌توان رایج‌ترین و مهم‌ترین آن‌ها را نام‌برد که عبارت‌اند از، **نحوه‌ی یادگیری** که معمولاً این مسائل را در دو فرم با نظارت و بدون نظارت دسته‌بندی می‌کند و **خواسته‌ی مسئله** که به‌طور 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 داده‌های جدید استفاده می‌کند. این تابع را در ادامه با $y(x)$ نشان می‌دهیم.

به‌طور کلی در یک مسئله‌ی یادگیری^{۳۵} در صورتی که خواسته‌ی مسئله درون‌یابی^{۳۶} یا برون‌یابی^{۳۷} یک داده‌ی پیوسته باشد به آن یک مسئله‌ی رگرسیون^{۳۸} می‌گویند و در صورتی که هدف پیش‌بینی یک کلاس گسسته و محدود برای داده‌های جدید باشد به آن کلاس‌بندی^{۳۹} می‌گویند، یا به تفسیری دیگر از کتاب پترن ریکاگنیشن:

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 به‌عنوان مدلی برای درک بهتر حافظه‌ی انسان نیز استفاده می‌شوند. [۹]

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. [۶، ص-۱۱ و ۱۲]

۳-۵-۲- ماشین بلترمن

هینتون^{۳۲} و سجنوفسکی^{۳۳} در سال ۱۹۸۵ یک قانون یادگیری جامع برای شبکه‌های احتمالاتی متقارن ابداع کرده (بخش ۵-۳) و بر اساس آن ماشین بلترمن را طراحی کردند.

ماشین بلترمن 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.

قابل‌خاطر نشان کردن است که الگوریتم اولیه‌ی بلترمن به‌خاطر نیاز به محاسبه‌های طاقت‌فرسا بر روی متغیرهای احتمالاتی بسیار کند است اما مدل قطعی میدان میانگین^{۳۴} سرعت یادگیری این ماشین را به‌طور چشم‌گیری افزایش می‌دهد. [۶، ص-۱۶۳]

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

^{۳۵} Learning Problem

^{۳۶} Interpolation

^{۳۷} Extrapolation

^{۳۸} Regression

^{۳۹} Classification

^{۳۱} local minima

^{۳۲} Hinton

^{۳۳} Sejnowski

^{۳۴} mean field

در صورتی که هدف یک فرایند، انتساب داده های ورودی به تعداد متناهی ای از کلاس های گسسته باشد به آن کار کلاس بندی می گوئیم و اگر خروجی (یعنی خروجی تابع $y(x)$) شامل یک یا چند داده پیوسته باشد در این صورت به این کار رگرسیون می گوئیم. [۳]

در نتیجه این دسته بندی بر اساس خروجی تابع y است.

۴-۲- نحوه یادگیری

فرض کنید یک مجموعه داده $X = x_1, x_2, \dots, x_N$ در اختیار شما قرار گرفته و از شما می خواهند که با استفاده از آن یک روش و یا الگو ارائه دهید که در صورت مواجه با داده ی جدید \hat{x} با آن الگو بتوان یک ویژگی از \hat{x} به اسم \hat{t} را با تابع $y(\hat{x})$ پیش بینی کرد. همان طور که از جمله ی 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 مجموعه ی تعلیم 40 نام دارد یک بردار $t = t_1, t_2, \dots, t_N$ به نام بردار هدف 41 از مقدار ویژگی مورد نظریه ازای ورودی های متناظر سوال به ما داده می شود، و هدف آن است که با یافتن یک الگو در این ۲ مجموعه $y(x)$ را شکل دهیم و این ویژگی را برای ورودی هایی که \hat{t} به ما داده نشده با $y(\hat{x})$ پیش بینی کنیم، به یادگیری این دسته از مسائل به علت راهنمایی هایی که در مرحله ی یادگیری به الگوریتم داده می شود، یادگیری با نظارت 42 گفته می شود.

اما در بعضی دیگر از مسائل بردار t به الگوریتم داده نمی شود، با این که ممکن است در نگاه اول عجیب به نظر برسد که الگوریتم باید بدون داشتن هیچ ایده ای از خواسته ی مسئله به دنبال ویژگی ای در مسئله بگردد، اما در ادامه می بینیم در واقعیت بسیار با این گونه مسائل روبه رو می شویم، برای مثال وقتی که برای اولین بار کسی با چند دسته از اشیاء جدید روبه رو می شود، بدون آن که کسی به او بگوید این اشیاء با هم تفاوت دارند، فرد از روی تفاوت هایی مانند شکل، اندازه و ... آن ها را در گروه هایی دسته بندی می کند و برای هر گروه مفهومی در ذهن خود در نظر می گیرد، از آن جا که این دسته از مسائل هیچ گونه راهنمایی ای در مرحله ی یادگیری دریافت نمی کنند، یادگیری این دسته مسائل را یادگیری بدون نظارت می نامند.

۵- یادگیری

بر اهمیت ترین ویژگی شبکه های عصبی، توانایی یادگیری و بهبود عملکرد آن ها به وسیله یادگیری است. در واقع اتصالات بین

عصب ها و شیوه ی فعالیت آن ها، تابع $y(x)$ که در قسمت قبل مطرح شد را می سازند، پس در شبکه های عصبی مرحله ی یادگیری که مرحله ی تولید y است، با تعیین وزن اتصالات بین عصب ها انجام می شود. تعریف دقیق یادگیری در شبکه های عصبی را می توان به صورت زیر دانست:

یادگیری فرایندی است که در آن متغیرهای آزاد یک شبکه ی عصبی (یعنی وزن اتصالات و ...) در جریان تحریک شدن به وسیله ی محیطی که شبکه در آن قرار دارد مقدار می گیرند. نوع یادگیری، تغییراتی که در این متغیرها رخ می دهد را تعیین می کند. [۱، ص- ۵۰]

با توجه به تعریف بالا می توان گفت که یادگیری در شبکه های عصبی از دنباله ی 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 مرحله ی قبل.

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

۱. تصحیح خطا
۲. یادگیری هببین
۳. یادگیری رقابتی
۴. یادگیری بر مبنای حافظه
۵. یادگیری بتلزنمن

در بخش های بعدی درباره ی هر یک از این روش های یادگیری توضیحی مختصر داده می شود، برای یادگیری بیشتر در این زمینه می توانید به منابع ذکر شده در قسمت پایانی رجوع کنید.

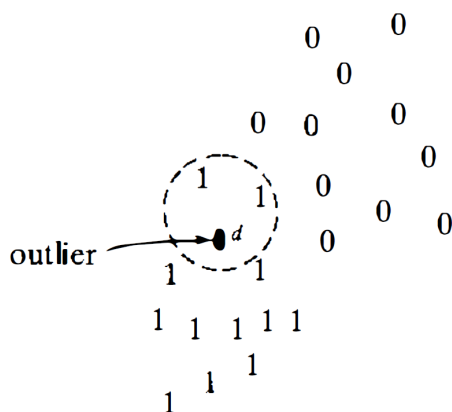
۵-۱- قواعد تصحیح خطا

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

⁴⁰ Training set

⁴¹ Target vector

⁴² Supervised learning



شکل ۹: محدوده‌ی درون خطچین شامل دو نقطه متعلق به کلاس ۱ و یک داده‌ی پرت با مقدار ۰ است، الگوریتم k نزدیک‌ترین همسایه ($k=3$) مقدار به طور شهودی صحیح ۱ را به نقطه‌ی d نسبت می‌دهد، در حالی که d به داده‌ی پرت ۰ نزدیک‌تر است. (کتاب شبکه‌های عصبی [۱]، ص-۵۵)

ساده مبتنی بر حافظه قانون نزدیک‌ترین همسایه^{۴۳} است. در این جا همسایگی \hat{x} ، آزمایشی است که در نزدیک‌ترین فاصله از آن قرار دارد یعنی آزمایشی که بردار ورودی آن تا بردار \hat{x} کمترین فاصله‌ی اقلیدسی را داراست:

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

مقداری که به این داده یعنی x'_N نسبت دارد به عنوان مقدار خروجی \hat{x} گزارش می‌شود، می‌توان ثابت کرد در صورت برقرار بودن چند شرط بر روی داده‌ها، 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 نزدیک‌ترین همسایه است. در این روش همسایگی \hat{x} ، k آزمایشی است که در نزدیک‌ترین 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.

الگوریتم‌های 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.

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 عصب‌ها به منظور بهبود دادن خروجی است. این تغییرات طراحی شده‌اند 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:

$$\xi(n) = \frac{1}{\gamma} e_k^2(n)$$

که در اینجا $e_k(n)$ نشان دهنده‌ی مقدار خطا در مرحله‌ی n ام است. می‌توان 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 ورودی‌ها و مقدار خطا متناسب است.

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

$$\omega_{k,j}(n+1) = \omega_{k,j}(n) + \Delta\omega_{k,j}(n)$$

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 عصب‌های مجاور به دست می‌آید و حالت‌های پیچیده‌تر را در نظر نمی‌گیرد.

۲-۵ یادگیری بر مبنای حافظه

در یادگیری بر پایه‌ی حافظه، بیشتر تجربیات گذشته به صورت واضح در یک حافظه‌ی رده‌بندی شده به صورت ورودی-خروجی یعنی $\{(X_i, d_i)\}_{i=1}^N$ 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 \hat{x} (ورودی جدید) الگوریتم با بررسی داده‌های آزمایشی در همسایگی \hat{x} و آنالیز آن‌ها پاسخ خود را می‌دهد.

در تعریف بالا دو چیز نیاز به رفع ابهام دارد، یکی مفهوم همسایگی \hat{x} و دیگری روش اعمال قوانین یادگیری بر روی داده‌های آزمایشی در همسایگی \hat{x} است. برای مثال یک روش یادگیری

^{۴۳}Nearest neighbor rule

۵-۳- بلتزن (Boltzmann)

در سال ۱۹۸۶ هینتون و سجنوفسکی یک قانون یادگیری را برای شبکه‌های احتمالاتی متقارن ابداع کردند و نام آن را به خاطر توزیع بلتزن به افتخار طراح آن نام گذاری کردند. به شبکه‌های عصبی‌ای که بر پایه‌ی این الگوریتم کار می‌کنند ماشین‌های بلتزن (بخش ۳-۵-۲) می‌گویند.

در یک ماشین بلتزن 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 عصب‌ها 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 ΔE_k مقدار تغییرات انرژی بر اثر تغییر وضعیت عصب k را نشان می‌دهد.

عصب‌های ماشین بلتزن را می‌توان به دو دسته‌ی پنهان و آشکار تقسیم کرد، این ماشین در دو حالت مختلف عمل می‌کند:

۱. حالتی که وضعیت عصب‌های آشکار از سوی محیط تحمیل می‌شود.

۲. حالتی که عصب‌های آشکار آزادانه تغییر می‌کنند.

که در هر دو این حالات وضعیت عصب‌های پنهان به صورت آزاد می‌تواند تغییر کند. فرض کنید $\rho_{i,j}^+$ مقدار هم بستگی وضعیت‌های دو عصب i و j و میانگین گرفته شده در تمام 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 متغیرهای بالا قبل تر تعریف شده‌اند.

۵-۴- هبین (Hebbian)

فرضیه‌ی هب یکی از قدیمی‌ترین و مشهورترین قانون‌های 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 عصب‌ها 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 هم‌زمان 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 عصب‌ها و بهینه سازی آنها استفاده می‌کند.

از تعریف اولیه، قانون دوم نوشته شده به دست نمی‌آید در واقع مدل‌های 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 هم‌زمانی‌ها 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.

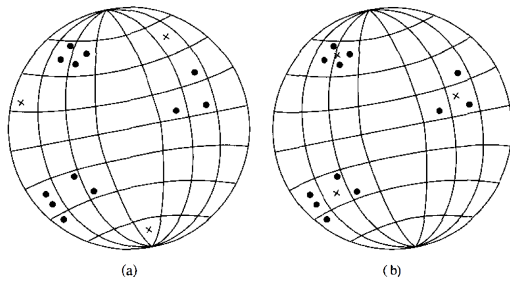
۵-۵- آموزش رقابتی

مشابه اسم این یادگیری، عصب‌های شبکه‌های عصبی در این روش با یکدیگر سر فعال شدن رقابت می‌کنند، برعکس روش‌های 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 شبکه‌ها را برای کشف ویژگی‌های مهم آماری برای دسته بندی داده‌ها مناسب می‌سازد. قواعد این روش را در سه عنصر می‌توان خلاصه کرد:

۱. یک مجموعه از عصب‌ها که کاملاً مشابه‌اند، به جز در وزن اتصالات که به صورت تصادفی توزیع شده‌اند.

۲. یک محدودیت بر روی قدرت عصب‌ها.

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



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])

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.

۶-۲- خوشه سازی و دسته بندی

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

۶-۲-۱- استخراج اطلاعات

انسان ها در برابر حجم زیاد داده ها از ماشین ها عقب ماندند، علاوه بر برتری هایی که در بالا ذکر شد، گاهی ماشین ها نظمی در داده های پیدا می کنند که انسان به طور شهودی قابلیت درک آن را ندارد.

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^{۴۴} 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 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 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 آن 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. 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

^{۴۴}Local induced field

۶-۲-۲- تحلیل اطلاعات

به طور مشابه کامپیوترها در این زمینه هم ثابت کردند که در بعضی از موارد از انسان ها سریع ترند، بسیاری از شرکت های اقتصادی بر پایه ی این برنامه ها سرمایه گذاری های خود را انجام می دهند.

۶-۳- تقریب تابع ها

محاسبه ی بعضی از توابع به راحتی امکان پذیر نیست. تقریب تابع های پیچیده را به عنوان یکی دیگر از کاربرد های الگوریتم های ماشین لرنینگ می توان نام برد.

۶-۴- پیش بینی و حدس

پیش بینی وضع هوا، بازار، حدس زدن نتیجه ی یک رخداد و ... از کاربرد های دیگر شبکه های عصبی است، اکثر این موارد حتی به وسیله ی مغز انسان هم قابل پیش بینی نیستند.

۶-۵- بهینه سازی 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 ماشین های یادگیری می توانند 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.

مراجع

- [1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., New Jersey, Prentice-Hall, 1999.
- [2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., New Jersey, Prentice-Hall, 2010
- [3] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st 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
- [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