پروژه ۲۰: ایجاد یک سیستم خبره برای تشخیص انواعی از کمخونی، با هدف آشنایی با سیستمهای شبکه عصبی مصنوعی

مقدمه:

کمخونی (آنمی) به صورت کاهش تعداد گلبولهای قرمز یا کاهش مقدار هموگلوبین یا کاهش حجم کل گلبولهای قرمز به حجم خون تعریف می‌شود. کمخونی یک بیماری نیست، بلکه علامتی برای تعداد زیادی از بیماریهاست. کمخونی هر چند در حالتهای خفیف یا مواردی که در زمان طولانی و به کندی ایجاد شده است ممکن است علائم واضحی نداشته باشد ولی در غیر از این موارد علانم بالینی گسترده و بسیار زیادی دارد که از حوصله این بحث خارج است.

در یک آزمایش شمارش سلولهای خون (CBC) بر اساس مقدار هموگلوبین (HGB) یا تعداد گلبولهای قرمز (Red Blood Cell - RBC) می‌توان به کمخونی پی‌برد. آزمایش شمارش سلولهای خون توسط دستگاه شمارنده سلولها (Cell Counter) انجام می‌شود. این دستگاه به این شکلی عمل می‌کند که مقدار کمی از خون لخته نشده بیمار در ظرفهای مخصوص نمونه آن قرار داده می شود و دستگاه به صورت کاملا اتوماتیک این نمونه را بررسی و گزارش نتایج را به صورت چاپ شده یا منتقل شده به رایانه تحویل می‌دهد. در گزارشهای این دستگاه‌ها حتی در موارد قدیمی معمولا بیش از ۱۰ شاخص خونی، اندازه گیری شده است. از موارد معمول این شاخصها تعداد گلبولهای قرمز (RBC)، تعداد گلبولهای سفید (WBC)، تعداد پلاکتها (PLT)، هموگلوبین (HGB)، هماتوکریت(نسبت حجم گلبولهای قرمز به حجم خون) (Hematocrit - HCT)، شاخص اندازه سلولها (Mean Corpuscular Volume - MCV)، شاخصهای رنگ سلول (MCHC و MCH) و شاخص پرکندگی اندازه گلبولهای قرمز (Red cell Distribution width - RDW) می‌توان نام برد.

تالاسمی یک بیماری کمخونی مادرزادی است که در این بیماران گلبولهای قرمز لیز (شکسته) می‌شوند و در نتیجه در خون این بیماران گلبولهای قرمز به صورت کوچکتر و کم رنگتر از حال طبیعی دیده می‌شوند. همچنین از نشانه‌های آن خونسازی غیر موثر است. تالاسمی در حقیقت مجموعه‌ای از بیماریها است و به گروههای زیادی تقسیم می شود. از نظر بالینی آن را به دو دسته تالاسمی ماژور و مینور تقسیم می‌کنند. تالاسمی ماژور علائم بالینی مشخصی دارد و مشکلات نسبتا زیاد بیماران، باعث تشخیص آنها در همان سالهای اول زندگی می‌شود. اما در تالاسمی مینور بیمار تقریبا هیچ شکایتی ندارد و به جز علانم آزمایشگاهی، این بیماری نشانه بالینی خاصی ندارد. از این جهت حتی آن را بیماری نمی‌نامند و اصولا این افراد نیازی به درمان ندارند. با این وجود باید تالاسمی مینور را تشخیص داد، چرا که در صورت ازدواج مرد و زنی که هر دو تالاسمی مینور دارند، فرزند آنها ممکن است مبتلا به تالاسمی ماژور به دنیا بیاید. همین امر باعث شده است که در کشور ما از کارهای که به صورت قانونی در هنگام ازدواج باید صورت گیرد بررسی زوجین از نظر تالاسمی مینور است. مراحل تشخیص به این شکل است که افرادی که در آزمایش شمارش سلولهای خونی (CBC) آنها، گلبولهای قرمز کوچک و کمرنگ وجود دارد که این امر با کاهش هموگلوبین

(Hg) و شاخص اندازه سلول (Mean Corpuscular Volume -MCV) مشخص می‌شود، با آزمایشهای اختصاصیتری از قبیل الکتروفورزیس هموگلوبین بررسی شده و تشخیص نهایی به دست می‌آید.

نکته‌ای که در این جا باید به آن اشاره کرد این است که پس از مشخص شدن اندازه کوچک گلبولهای قرمز با توجه به شاخص MCV، چند تشخیص افتراقی وجود دارد. این کوچکی ممکن است به علت تالاسمی مینور یا کمخونی فقر آهن یا بیماریهای مزمن باشد. در مورد بیماریهای مزمن معمولا مشکل چندانی نیست و این بیماران به خاطر علائم و تشخیص قبلی، معمولا به سادگی از بقیه بیماران جدا می شوند. اما در فردی که به عنوان مثال در هنگام ازدواج مشخص می شود که MCV پایینی دارد، باید تعیین نمود که تالاسمی مینور دارد یا فقر آهن. برای بدست آوردن این تشخیص می‌توان از آزمایشهای اختصاصی تالاسمی مینور مانند الکتروفورزیس هموگلوبین یا از آزمایشهای اختصاصی آهن مانند میزان فریتین و آهن خون استفاده نمود.

نکته‌ای قابل توجه این است که از مدتها پیش مشخص شده بود که بر اساس شاخصهای موجود در گزارش CBC تا حدی می‌توان در بیماری که MCV پایینی دارد، بیماران تالاسمی مینور و فقر آهن را از هم جدا نمود.

همچنین به خاطر داشته باشید که آزمایش شمارش سلولهای خون یک آزمایش ارزان، ساده و سریع است و دستگاه شمارنده سلولها حتی در آزمایشگاههای کوچک در شهرهای کوچک نیز موجود است. از طرف دیگر آزمایشهای اختصاصی تالاسمی و فقر آهن که پس از بدست آوردن گزارش شمارش سلولهای خون انجام می‌شوند، آزمایشهای نسبتا گرانی هستند که در آزمایشگاههای مجهز قابل انجام می‌باشند.

عوامل ذکر شده باعث شده است که محققان تلاشهایی را در جهت پیدا نمودن راهی برای افتراق تالاسمی مینور و فقر آهن بر اساس گزارش CBC شروع نمایند. این تلاشها در ابتدا باعث بدست آوردن مدلهای ریاضی ساده شد.

تعدادی از این مدلها را می‌توانید در جدول زیر رویت نمایید.

Discriminant function (DF)

	Formulae	Reference
Green's DF	$= MCV^2 \times RDW\text{-}CV / 100 / HGB$	Green, et al.[3]
Shine's DF	$= MCV^2 \times MCH / 1000$	Shine, et al.[4]
Bruno's DF	$= 0.096MCV + 0.415RDW\text{-}CV - 0.139RBC$	Bruno, et al.[5]
England's DF	$= MCV - 5 \times HGB - RBC$	England, et al.[1,2]
Mentzer's DF	$= MCV / RBC$	Mentzer[6]
Srivastava's DF	$= MCH / RBC$	Srivastava, et al.[7]

مهمترین اشکالی که این مدلهای ریاضی داشتند و باعث شد که پزشکان باز هم به آزمایشهای اختصاصی روی آورند عدم کفایت حساسیت (Sensitivity) و اختصاصی بودن (Specificity) این مدلها بود.

در سال ۱۹۹۶ در Brindorf Nt و همکارنش در تلاشهایشان برای پیدا نمودن راه حلی برای مشکل فوق، یک سیستم خبره مرکب طراحی نمودند. در این سیستم از مفاهیم سیستمهای خبره بر پایه قوانین (Rule-

based) بعلاوه مفاهیم شبکه عصبی مصنوعی استفاده شده بود. این شبکه عصبی ۳ لایه داشت و ورودیهای آن عبارت بودند از هماتوکریت (HCT)، MCV و RDW. همانطور که می‌دانید همه این ورودیها از گزارش CBC بدست می‌آید. خروجیهای مدل عبارت بود از دسته بندی بیماران بر اساس دسته سالم و سه دسته کمخونی با سلول کوچک و کمرنگ که عبارتند از آنمی فقر آهن، اشکالات ساختمانی هموگلوبین که در راس آن تالاسمی قرار می‌گیرد و کمخونی ناشی از بیماریهای مزمن. این سیستم بعد از آموزش توانست در مرحله بررسی صحت، ۴۷۳ بیمار را با صحت ۹۶,۵ درصد تشخیص دهد.

همانطور که گفته شد ورودیهای سیستم فوق همه در یک گزارش CBC موجود هستند و دستگاه شمارنده سلول نیز هم خود رایانه داخلی دارد و هم می‌تواند اطلاعات و گزارشهای خود را به یک رایانه دیگر انتقال دهد. پس اگر سیستمی شبیه آنچه در بالا توضیح داده شد را در این رایانه‌های قرار دهیم، گزارش نهایی می‌تواند علاوه بر نتایج معمول CBC ، در مورد علت کمخونی نیز گزارش قابل قبولی دهد. در این شرایط در موارد پایش بیماران تالاسمی که از طریق آزمایش CBC انجام می‌شود در صورت پایین بودن MCV نیاز به انجام آزمایشهای اختصاصی گران قیمت و زمانبر، نخواهد بود.

هدف:

هدف این پروژه آشنایی اولیه محققان و دانشجویان بویژه محققان و دانشجویان گروه پزشکی با مفاهیم شکبه عصبی مصنوعی است. در این راستا در ادامه دو فصل خواهیم داشت. در فصل اول توضیحی در مورد مفاهیم و طرز کار شبکه‌های عصبی ارائه می‌شود و سپس در فصل دوم با *دیدگاهی ساده* تهیه و طراحی یک سیستم شبکه عصبی را برای افتراق کمخونی، شبیه سازی می‌کنیم و یک شبکه عصبی مصنوعی را در برنامه MATLAB پیاده‌سازی می‌نماییم.

فصل اول:
ماهیت و طرز کار شبکه های عصبی

تجسم سناریویی کار با شبکه عصبی:

۱- تجسم کنید که شما روی فرمولاسیون یک شکل دارویی خاص کار می‌کنید فرض کنید که اطلاعات مربوط به چندین نمونه فرمول شناخته شده، از قبیل غلظت اجزاء، زمان فرایندهای مختلف فرمولاسیون، و پارامترهای فیزیکی از قبیل دما و PH را به یک برنامه کامپیوتری می‌دهید سپس زمان پایداری اندازه‌گیری شده برای نمونه‌ها را هم به این برنامه می‌دهید شما هیچ اطلاع یا فرضیه‌ای در مورد نحوه تاثیر اطلاعات و پارامترهای داده شده بر روی پایداری اندازه‌گیری شده ندارید. برنامه مذکور مدتی "فکر می‌کند"، داده‌های سیستم شما را تجزیه و تحلیل می‌کند، روابط پیچیده بین پارامترها را یافته و مدل ریاضی آنها را می‌سازد، به طوریکه شما می‌توانید اطلاعات مربوط به یک نمونه جدید را به آن وارد کنید و برنامه پایداری آن را با دقت قابل قبولی برای شما پیش‌بینی کند.

۲- تجسم کنید که مصرف یک دارو در شرایط خاص ایجاد مسمویت می‌کند احتمالا پارامترهای مختلفی در این امر تاثیر دارند از جمله سن بیمار، جنسیت بیمار، میزان داروی مصرف شده، زمان و نوع مصرف آن ، و ... شما این پارامترها را برای چندین بیمار به برنامه می‌دهید و وضعیت نهایی از قبیل ایجاد یا عدم ایجاد مسمومیت و یا شدت آن را به برنامه می دهید. برنامه مذکور مدتی "فکر می‌کند"، دادههای سیستم شما را تجزیه و تحلیل می‌کند، و روابط پیچیده بین پارامترها را یافته و مدل ریاضی آنها را می‌سازد به طوری که شما می‌توانید با وارد کردن اطلاعات مربوط به بیمار جدید وضعیت آتی وی از قبیل بروز یا عدم بروز مسمویت و نیز شدت آن را با دقت قابل قبول پیشگویی کنید.

چه اتفاقی افتاد؟

برنامه کامپیوتری مذکور روابط پیچیده میان چند عامل مختلف را با پایداری محصول "حس" و "کشف" کرد، بدون اینکه شما حدس یا فرضیه‌ای در مورد ماهیت این رابطه داشته باشید. هیچ اثری از تستهای پیچیده آماری نیست حتی مجبور نشدید از داده‌های ورودی میانگین بگیرید... .تجسم شما یک واقعیت است، که به سادگی قابل تجربه و در دسترس شما است. پیش از اینکه بیشتر درباره جزئیات این قبیل برنامه‌ها بدانید، نام آنها را به خاطر بسپرید. نام این سیستم :شبکه‌های عصبی مصنوعی یا Artificial Neural Networks " است.

اشکالات روشهای کلاسیک تجزیه و تحلیل داده‌ها

تقریبا تمام محققینی که با آنالیز داده‌ها سر و کار دارند به اشکالات سیستم‌های کلاسیک آنالیز واقف هستند:

۱- لزوم وجود تئوری و فرضیه اولیه: تقریبا در کلیه روشهای کلاسیک آماری حدس یا فرضیه‌ای در مورد داده‌ها موجود است و روشهای آماری تنها وجود یا عدم وجود آن فرضیه را تائید یا تکذیب می‌کنند. این مسئله برای محققین به شکل عادتی درآمده است. اما چرا نباید سیستم آنالیز داده‌ها، خود، داده‌ها را تجزیه و تحلیل کنند و روابط بین آنها را بیابد؟

همانطور که مغز ما با دریافت داده‌های مربوط به چند مثال از یک دست، دست به تعمیم می‌زند و قانونی برای آنها می‌یابد. در واقع مرحله تفکر و استنتاج در روشهای کلاسیک یک مرحله نظری است که در ذهن محقق

انجام می‌شـــود. اما آیا می‌توان این مرحله را به نوعی اتوماتیک کرد و از قدرت پردازش کامپیوترها در آن کمک گرفت؟ به این معنی که کامپیوتر به جای اینکه تنها هضـــم و تبدیل داده‌ها را انجام دهد، خود ارتباطات و فرضیه‌های مختلف در مورد داده‌ها را طرح و آزمایش کند؟

۲- حساسیت به نویز و خطا: روشهای کلاسیک به شدت به وجود خطا و نویز در داده‌ها حساسند و این نکته برای کلیه محققین به شکل عادتی درآمده که اگر نویز از حدی بیشتر باشد دیگر نخواهند توانست با داده‌ها کار کنند چرا باید این چنین با شد؟ مغز ما می‌تواند صدای صحبت کردن یک شخص را در میان صدها صدای دیگر به دقت دنبال کند، چرا سیستم آنالیز داده‌ها نباید بتواند طیف وجود یک ماده را در میان طیف دهها ماده دیگر که با روی هم همپوشانی دارند جدا کند؟

۳- نیاز به کامل بودن داده‌ها: برای محققین این نکته مسـلم اسـت که پارامترهای مربوط به یک نمونه باید کامل با شد. یعنی اگر یکی از پارامترهای یک نمونه ناقص با شد (Missing Data) آن نمونه در آنالیز اسـتفاده نخواهد شـد. در صـورتی که اگر سـیسـتم به خطا و نویز بسـیار مقاوم باشـد می‌تواند از بقیه پارامترها اسـتفاده لازم را بکند و به جای پارامتر ناقص یک میزان دلخواه یا صـفر بگذارد، این خاصـیت، بسیاری از داده‌های بی‌استفاده در سیستم‌های فعلی را قابل استفاده می کند.

۴- سیستم‌های کلاسیک از درک، تعقیب، و ثبت روابط میان داده‌های متعدد به شدت عاجزند. به عنوان مثال، اثری مانند ریسـک بروز حمله قلبی را می‌تواند به بیش از بیسـت عامل مختلف ربط داد. اما حتی اگر این عوامل در نمونه‌ها اندازه‌گیری و ثبت شده باشند، نتیجه آنالیز آنها توسط سیستمهای کلاسیک تقریبا بی‌نتیجه خواهد بود به جز در مورد عواملی که سهم بزرگی در اثر مذکور دارند.

۵- در کلیه سیستم‌های کلاسیک، از ستونهای پارامترهای مربوط به داده‌ها، شاخص‌های مختلف آماری اسـتخراج می‌گرددو پس از آن دیگر کاری با تک تک نمونه‌ها نداریم. این طرز کار سـبب می‌شـود که با هم بودن پارامترهای مختلف در یک نمونه نادیده گرفته شـود بنابراین سـیسـتم آنالیز ما این (برهم کنش) میان پارامترها را نادیده گرفته از حس و ثبت آن عاجز است. بنابراین کلیه پیچیدگیهای مربوطه (که به شکل غیر خطی بودند ریاضی در مدل ما هستند) بدون درک باقی مانده و به شکل نوعی نویز یا خطا عمل می‌کنند و به همین دلیل سیستم آنالیز ما در تجزیه و تحلیل مدل ناتوان می‌ماند این درست شبیه این است که برای اندازه‌گیری تغییرات میزان یک عنصر در بافتی خاص، سلولهای آن را آسیاب کنیم و میزان عنصـر مذکور را در شـیرابه به دسـت آمده اندازه‌گیری کنیم. نتیجه این خواهد بود که تسـتهای آماری کلاسـیک هر گونه تغییر (افزایش یا کاهش) معنی‌دار آن عنصـر در بافت را مردود می‌شناسد. ممکن است این عنصر واقعا در سلول‌هایی از بافت به طور شاخصی زیاد یا کم شده است اما از آنجائی که میزان بسیار بیشتری (مثلا هزار برابر) از آن در خون یا مایع میان بافتی موجود است عمل هوموژن کردن و آسیاب کردن بافت، سبب می شود که تغییرات ظریف غلظت سلولی عنصر مذکور در زیر واریانس طبیعی تغییرات غلظت خونی یا میان بافتی آن پنهان شود. درصورتی که اگر ممکن بود که سلولها و اندامکهای سلولی خاص را جدا کرده و اندازه‌گیری را روی آنها بطور جداگانه انجام دهیم، آنگاه تغییرات ظریف غلظتی عنصر مذکور قابل ثبت بودند. هنگامی که شما با سیستمهای کلاسیک آنالیز دادها کار می‌کنید همین عمل آسـیاب کردن را روی دادهای خود انجام می‌دهید. چه بسـا محققین وجود اثری را در مدل خود کاملا حس می‌کنند (مثلا تاثیر دارو در بهبودی وضـعیت بیمار بطور بالینی

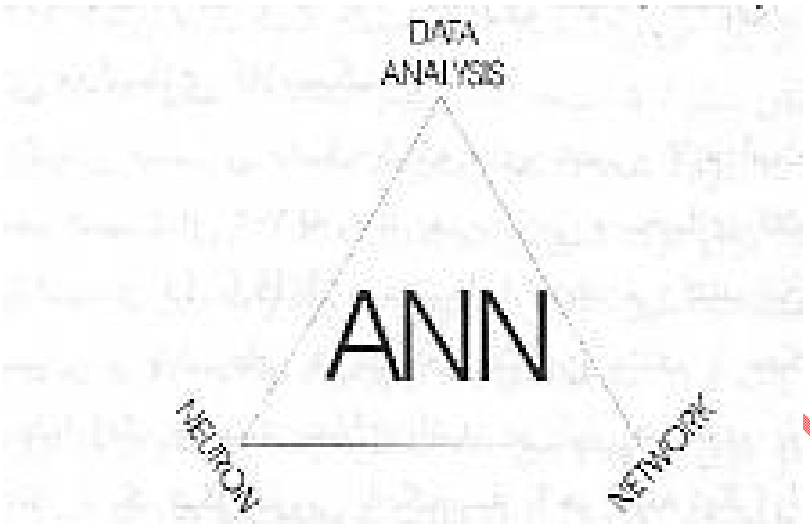
مطمئن هستند) اما در دام آنالیز کلاسیک گیر کرده و قادر به اثبات اثر نیستند. گاهی با انجام تستهای پیچیده‌تر یا با طراحی (Setup) آزمایشهای جدید می‌توانند اثر را ثابت کنند اما اگر اثر کمی پیچیده و چند عاملی (Multi factorial) باشد چه بسا هرگز موفق به ثبت آن نشوند. هر جا اثرات غیر قابل اثبات و تفاوتهای بی‌معنی از لحاظ آمار کلاسیک وجود دارد، سیستم جدید آنالیز داده‌ها (شبکه‌های عصبی) امیدی تازه به محققین می‌دهد.

تاریخچه:

از حدود ۱۹۴۰ بطور همزمان اما جداگانه، از سویی نوروفیزیولوژیستها سعی می‌کردند سیستم یادگیری و تجزیه و تحلیل مغز را کشف کنند و از سویی ریاضی دانان تلاش می‌کرند تا مدلی ریاضی بسازند که قابلیت فراگیری و تجزیه و تحلیل عمومی مسائل را دارا باشد. از آن زمان بارها این اتفاق افتاد که ریاضدانان یافته‌های نوروفیزیولوژیستها را پیاده سازی کردند بدون اینکه بدانند چرا، و در عمل مشاهده کردند که سیستم پیاده شده فوق، کارایی شگفت‌انگیز سیستم طبیعی را دارد. تنها پس از آن ریاضیدانان توانستند منطق زیر بنایی سیستم طبیعی را درک کنند. اگر چه از همان ابتدا ریاضیدانان توانسته بودند مدل ریاضی یک سلول عصبی یا نورون را بسازند، اما تا حدود ۱۹۷۴ که دانش مربوط به نوع تصال این واحدهای شبه نورونی به یکدیگر تکامل لازم را نیافته بود، هنوز اثر معجزه آسای یادگیری مغز شبیه سازی نشده بود. حتی پس از آن نیز، تا حدود ۱۹۸۴ که نخستین سری کامپیوترهای IBM XT در دسترس عموم محققین قرار گرفت، هنوز جاذبه‌های کاربردی این سیستم شکوفا نشده بودند. امروزه برنامه‌های کاربردی متعددی در دسترس هستند که با این روش کار می‌کنند. اگر چه کاربرد این برنامه‌ها بویژه برای افراد عادی کمی مشکل است اما محققین روز به روز بیشتر و بیشتر آنها را به کار می‌گیرند. برای تجزیه و تحلیل این سیستم پیچیده بوسیله روش شبکه‌های عصبی، نیاز به دانش زیادی در مورد سیستم مورد مطالعه نمی‌باشد، چون عمل تجزیه و تحلیل و یادگیری در مغز شبکه اتفاق می افتد نه در مغز محقق، اما به هر حال بهره‌گیری از دانش کلی درباره‌طرز کار این شبکه‌ها برای کاربران آنها ضروری است، چرا که تنظیمات ساده و کلی در این برنامه‌ها وجود دارند که آگاهی از آنها برای ساختن یک مدل موفق ضروری است.

تعریف:

شبکه‌های عصبی در واقع مثلثی هستند که سه ضلع مفهومی دارند: ۱- سیستم تجزیه و تحلیل داده‌ها، ۲- نورون یا سلول عصبی، ۳- شبکه یا قانون کار گروهی نورونها (شکل ۱-۱)



شکل ۱-۱: مثلث مفهوم شبکه های عصبی

در یک تعریف کلاسیک، هایکین می‌گوید: شبکه عصبی عبارت است از مجموعه‌ای عظیم از پردازشگرهای موازی که استعداد ذاتی برای ذخیره اطلاعات تجربی و به کارگیری آن دارند و این شبکه دست کم از دو بابت شبیه مغز ا ست: ۱- مرحله‌ای مو سوم به یادگیری دارد ۲- وزنهای سیناپسی جهت ذخیره دانش به کار می‌روند. شاید در اینجا این تعریف کمی گنگ به نظر برسد، اما در قسمتهای بعدی جزئیات آن را به دقت مرور خواهیم کرد.

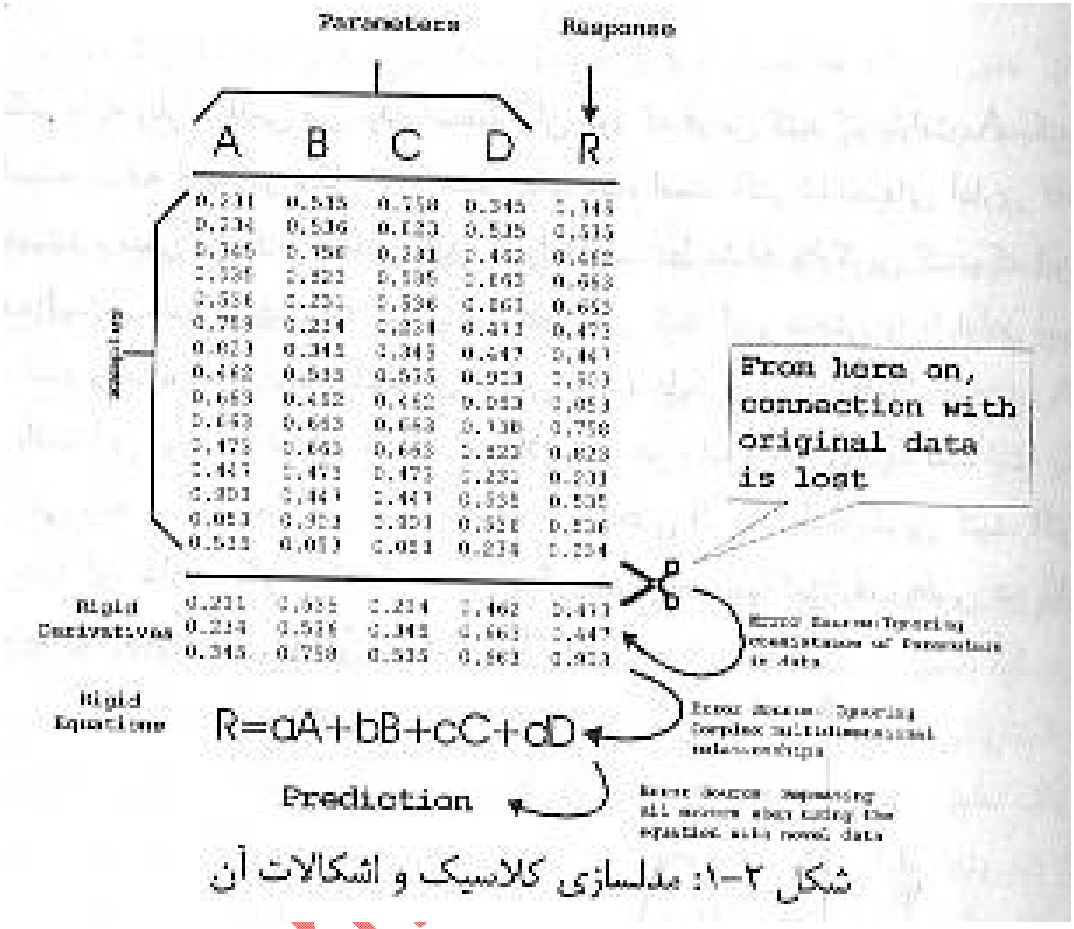
وظیفه شبکه‌های عصبی یادگیری است. تقریبا چیزی شبیه یادگیری یک کودک خردسال. یادگیر در شبکه‌های ع صبی رایج، به شکل Supervised یا یادگیر تحت نظارت ا ست. والدین ت صاویر حیوانات مختلف را به کودک نشـــان می‌دهند و نام هر کدام را به کودک می‌گویند. و ما روی یک حیوان، مثلا ســـگ، تمرکز می‌کنیم. کودک تصـــاویر انواع مختلف ســـگ را می‌بیند و در کنار اطلاعات ورودی (تصـــاویر و صـــدا) برای هر نمونه، به او گفته می شود که این اطلاعات مربوط به یک نوع سگ ه ستند یا خیر. بدون اینکه به او گفته شود، سی ستم مغز او اطلاعات ورودی را تجزیه و تحلیل می‌کند و به یافته‌هایی در زمینه هر یک از پارامترهای ورودی از قبیل ((رنگ، اندازه، صدا، داشتن پنجه، سم یا شاخ))می‌رسد. پس از مدتی او قادر خواهد بود یک نوع ((نوع جدید)) از سگ را قبلا هرگز ندیده ا ست شنا سایی کند. از آنجائیکه که در مورد هر نمونه جانور در مرحله یادگیری به کودک گفته شده که آیا سگ هست یا خیر، این نوع یادگیری، یادگیری تحت نظارت نامیده می‌شود. نوع دیگر یادگیری یعنی یادگیری بدون نظارت یا Unsupervised هم تو سط شبکه‌های ع صبی شبیه سازی شده ا ست و کمتر کاربرد دارد.

مدلسازی کلاسیک در مقایسه با مدلسازی شبکه عصبی:

الف: سناریوی مدلسازی کلاسیک

جهت درک ویژگیهای مدل سازی به کمک شبکه‌های ع صبی، لازم ا ست که ابتدا مدل سازی کلا سیک را تج سم کنیم. شکل ۲-۱ نحوه تعریف، بررسی و مدلسازی کلاسیک یک سیستم را مرور می‌کند: پارامترهای A,B, C, D سیستم ما را تعریف می‌کنند این پارامترها می توانند غلظت مواد شیمیایی و پارامترهای فیزیکی فرمولاسیون

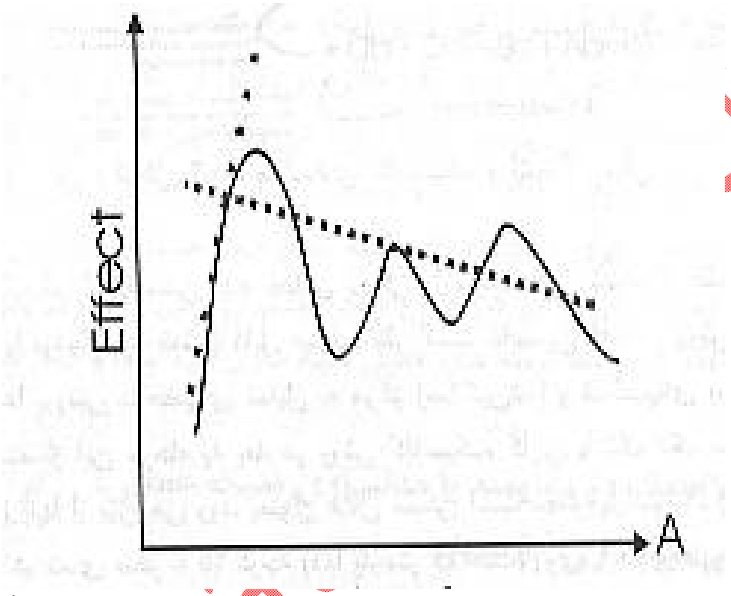
با شند یا ممکن ا ست پارامترهای فیزولوژیکی یا پاتولوژیک و ضعیت بیماران با شند. در کنار هر سری پارامتر که یک نمونه (فرمولا سیون خاص از یک شکل داوریی یا یک بیمار یا هر نمونه دیگر) را تعریف می‌کنند. یک پا سخ R وجود دارد که می‌تواند پایداری فرمول داوریی یا پارامتری نمایانگر وضعیت بیمار باشد.



شکل ۲-۱: مدلسازی کلاسیک و اشکالات آن

مدل سازی کلا سیک از نخ ستین قدم خطای بزرگی را مرتکب می شود که فقط در سیستمهای ساده (خطی یا نزدیک به خطی) قابل صرف نظر ا ست.نخستین قدم در روش کلا سیک برای برر سی دادها برر سی شاخ صهای تمایل به مرکز (میانگین، ...) و شاخصای پراکندگی (انحراف، معیار، ...) است. از این مرحله به بعد روش کلاسیک، کاری با تک تک نمونه‌ها نداریم و اهمیت فردی آنها از بین می‌رود به عنوان مثال ممکن اســـت مقدار پارامتر A (غلظت ماده شیمیایی A) در دمای ۰ تا ۱۵ درجه به بالا تاثیر منفی دا شته با شد، به این ترتیب نگاه همزمان به پارامترهای Aو B برای درک تاثیر آن دو بر روی R ضـــروری اســـت.در حالی که ما با گرفتن میانگین (یا دیگر مشتقات آماری) از کل ستونهای A وB و سایرین اثر همراهی مقادیر A و B را از صورت مسئله پاک کرده‌ایم در هیچ یک از رو شهای کلا سیک مدلسازی با داده‌های فردی (تک تک نمونه ها) کاری نداریم و این یک ا شکال مهم ا ست. **در واقع روش کلا سیک با عملی شبیه به هموژن کردن یا آ سیاب کردن دادها، پیچیدگی روابط آنها را محو می کند و به این دلیل از کشف این پیچیدگیها باز می‌ماند.**

این مطلب را به زبان ریاضی می‌توان این طور بیان کرد که فرض کنیم پارامتر A یک اثر غیر خطی است، مشابه آنچه در شکل ۳-۱ نشان داده شده است اکثر شاخصهای آماری کلاسیک، خطی هستند و سعی می‌کنند منحی تاثیر A را با یک خط مشابه جایگزین کنند، که این عملی بی‌فایده است. روشهای پیشرفته آماری نیز سعی می کنند. این منحنی را با توابع غیر خطی شناخته شده و ساده (مثلا توابع زنگوله‌ای گاوس یا ...) جایگزین کنند که اگر منحنی A خیلی پیچیده با شد این توابع نیز ناتوان هستند.به علاوه، در هر یک از این رو شها، شما یک ابتدا یک تابع را حدس می‌زنید و بعد سعی می‌کنید به کمک آن منحنی اثر A را جایگزین کنید. اگر موفق نشوید، چاره‌ای ندارید مگر اینکه باز حدس دیگری بزنید به طور کلی راه روشنی در برابر شما نیست و چه بسا هیچ حدس موفقی به ذهن شما نرسد.



_____ Experimental
.....Classical Modeling (Global)
.....Classical Modeling (Local)

شکل ۳-۱: ناتوانی مدلسازی کلاسیک در مدل کردن رفتارهای پیچیده غیر خطی

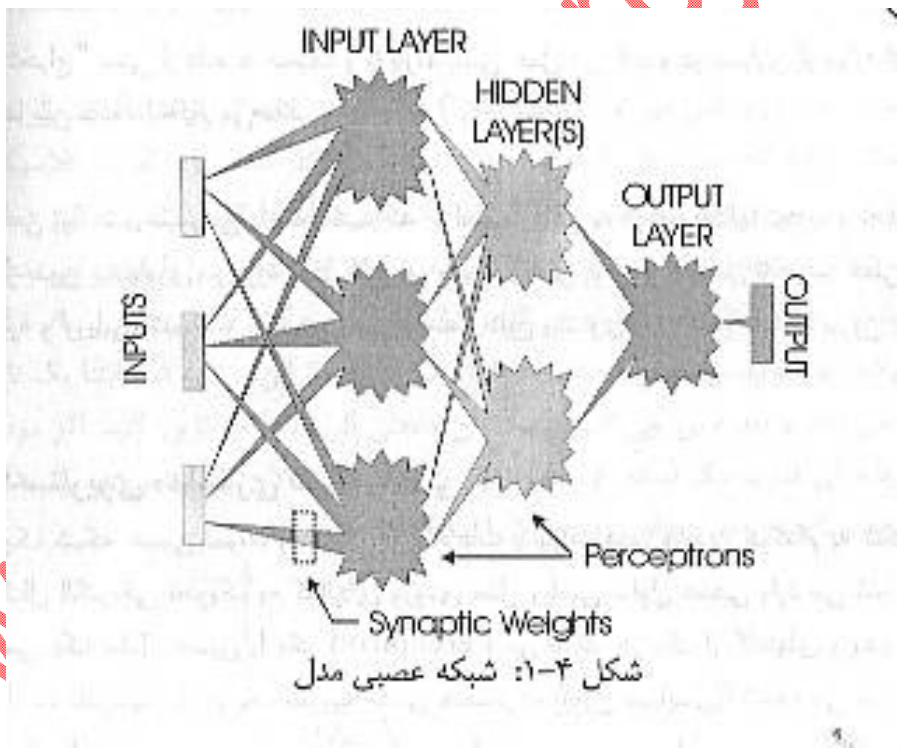
در نهایت نیز، در روش کلاسیک، شما یک معادله سیستم خواهید داشت که داده‌های جدید را بدون در نظر گرفتن اثر همراهی پارامترهایش با هم ا ستفاده می کنند و مجددا این خطا در پیشگویی اثر R توسط سیستم شما ناثیر خواهد داشت.بدین ترتیب سیستم کلاسیک در "استخراج" معنی از داده‌ها ضعیف و بازده پایین عمل می‌کند و در بسیاری از موارد از کشف روابط بین داده‌ها ناکام می‌ماند اگر می‌توانستیم سیستمی داشته با شیم که با اهمیت دادن به فرد فرد مثالها تجزیه و تحلیل کند و نیز بدون پیشداوری در مورد شکل تابع هر پارامتر (خطی بودن و یا شکل تابع غیر خطی)آن را ذخیره و ارزیابی کند. چنین سیستمی می‌توانست نتایج بیشتری را از عمق داده‌ها بیرون بکشد.

ب:سناریوی مدلسازی شبکه عصبی:

در یک شبکه عصبی نمونه، (شکال ۴-۱) اطلاعات و پارامترهای ورودی هر کدام به شکل یک سیگنال الکتریکی تحریک به کانالهای ورودی مدل ریاضی سلول عصبی وارد می‌شوند. مدل ریاضی یک سلول عصبی را یک

Perceptron می نامند. هر یک از کانالهای ورودی (شبیه اتصالات دندریتها) دارای یک ضریب عددی هستند که ((وزن سیناپسی)) نامیده می شود. شدت تحریک الکتریکی در این ضریب ضرب می شوند . به جسم سلولی رسد اگر مجموع تحریکات رسیده به جسم سلولی کافی باشد، نرون شکلیک می‌کند و در مسیرهای خروجی (نظیر آکسونها) جریان الکتریکی ثابتی را ایجاد می‌کند. تحریکات لایه ورودی سلولها به یک یا چند لایه واسط می‌رود که به نام لایه‌های مخفی(Hidden Layers) موسوم هستند ادامه جریان تحریکات در این لایه‌ها (توسط همان وزنهای سیناپسی)طوری هدایت می شود که پیچیدگیهای تاثیرات جریان ورودی را شبیه سازی می‌کند. سپس تحریکات به لایه خروجی می‌روند که هدف نهایی ما است.

اگر هدف شبکه عصبی پیشگویی کمی باشد، مجموع شدت تحریکات آخرین عصب خروجی، آن عدد خواهد بود اگر هدف شبکه عصبی طبقه‌بندی (Classification) باشد، فعالیت یا خاموش بودن نرونهای لایه‌آخر نمایانگر این امر خواهد بود مثلا شلیک نرون خروجی نشان‌دهنده حضور بیماری و خاموش‌بودن آن نشانه سلامت است. سیستم شبکه عصبی در فرایند یادگیری طوری وزنها سیناپسی تغییر می دهد که بتواند با هر سری تحریکات ورودی (یعنی داده‌های هر نمونه) جریان خروجی مناسب (یعنی همان پاسخ R) را ایجاد کند چگونگی ایجاد ریاضی این تغییر وزنها ظریفترین بخش مکانیسم عملکرد شبکه است که بعدها درباره آن بحث خواهیم کرد

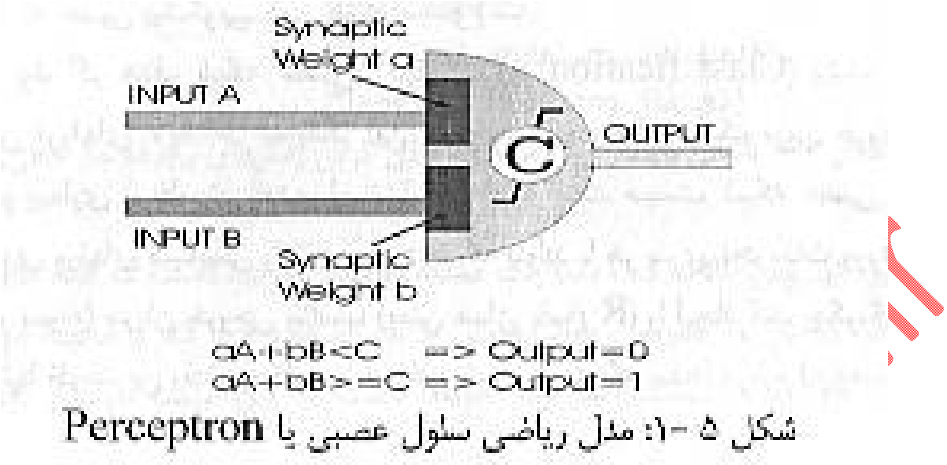


شکل ۴-۱: شبکه عصبی مدل

طرز کار مدل سلولی عصبی:

نمونه یک سلول عصبی مدل در شکل ۵-۱ نمایش داده شده است این مدل ریاضی که از روی الگوی ریاضی آن ساخته شده است Perceptron نامیده می شود خطوط ورودی (Input) سیگنالهای تحریکی یا مهاری را به جسم سلولی می‌آورد که همان پارامترهای تعریف کننده سیستم هستند مثلا فرض کنیم که غلظت یک ماده

0.6 Mol /Lit است این عدد یکی از پارامترهای تعریف کننده نمونه دارویی ما است پس این پارامتر به عنوان یک سیگنال الکتریکی با شدت 0.6 به یک کانال ورودی می‌رود.

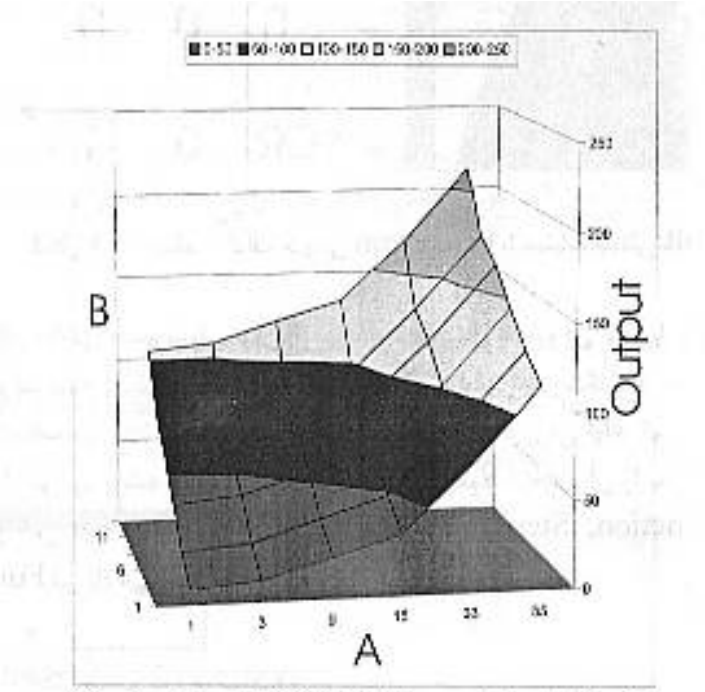


شکل ۵-۱: مدل ریاضی سلول عصبی یا Perceptron

در ابتدای هر کانال ورودی یک ضریب عددی وجود دارد که شدت تحریک در این عدد ضرب می شود و حاصل آنکه یک Weighted Input نامیده می‌شود اگر مثبت باشد یک سیگنال تحریکی و اگر منفی باشد یک سیگنال مهاری وارد بر جسم سلولی است. میزان کلیه این سیگنالهای تحریکی یا مهاری که از ورودیهای مختلف به جسم سلولی می‌رسند به صورتی خطی جمع می‌شوند (Linear combination Of Weighted Inputs) اگر این حاصل جمع از میزان آستانه یا Threshold کمتر باشد سلول عصبی خاموش می‌ماند و در غیر این صورت سلول شلیک می‌کند (Fire) و جریان الکتریکی ثابتی در خروجی (یا خروجیها) ایجاد می‌کند در واقع خروجی Perceptron از معادله ریاضی زیر تعیین می‌شود:

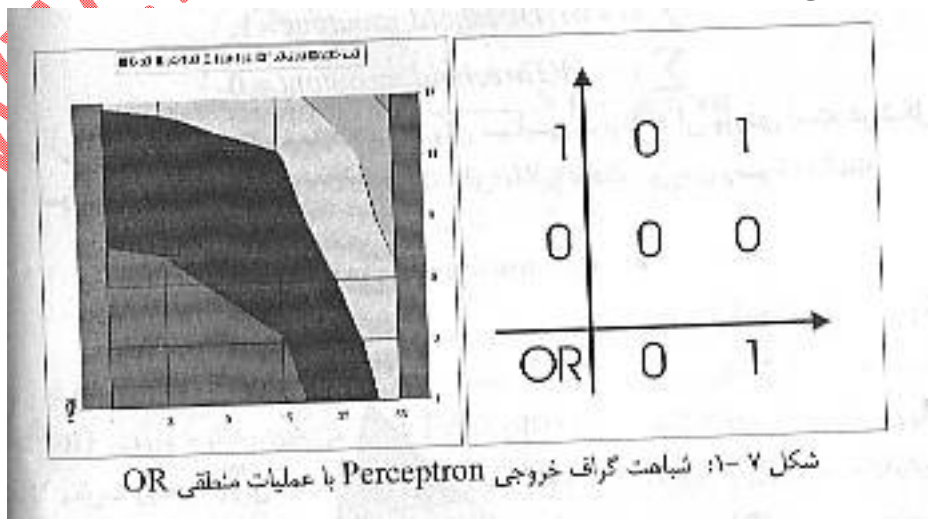
$$\sum xi \times wi > Threshold \Rightarrow output = 1$$
$$\sum xi \times wi < Threshold \Rightarrow output = 0$$

که در آن XI سیگنال ورودی شماره I و wi وزن سیناپسی مربوط به آن ورودی است. در شکل ۶-۱ نمودار خروجی یک Perceptron را به ازاء مقادیر مختلف ورودی رسم شده است.



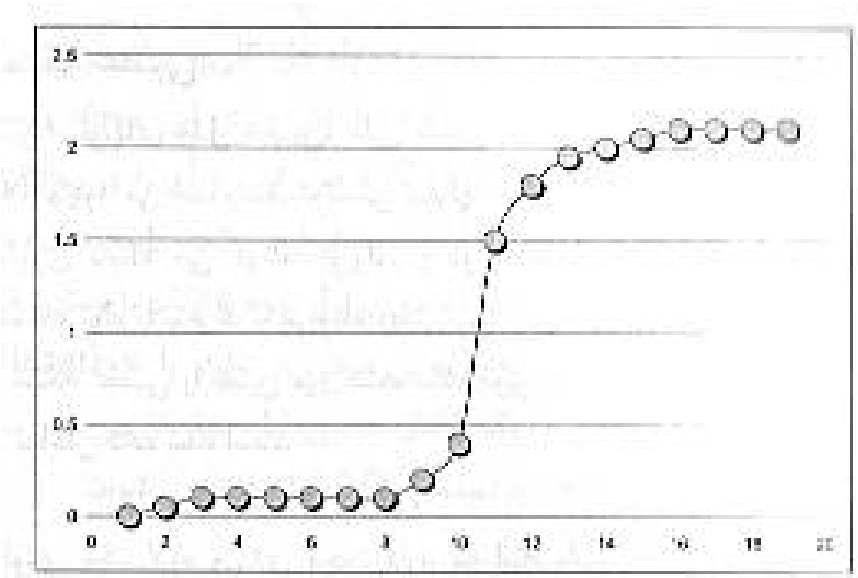
شکل ۶-۱: نمودار خروجی Perceptron با تغییر مقادیر A و B

اگر مطابق شـــکل ۷-۱: این گراف را از بالا و در کنار پارامترهای ورودی نگاه کنیم چیزی شـــبیه گراف دروازه منطقی "یا" (OR Logical Gate) می‌بینیم. به این ترتیب مقادیر خاصی از وزنهای سیناپسی می‌توانند سبب شوند که سلول عصبی ما روی ورودی آنالوگ (عددی) همان عملیات منطقی را انجام دهد که یک دروازه منطقی بر روی داده‌ای رقمی (دیجیتال= صـفر یا یک) انجام می‌دهد به همین ترتیب با دسـتکاری وزنه‌های سـیناپسـی می‌توان دروازه‌های منطقی AND وNOT را هم ایجاد کرد. عملیات منطقی پیچیده نیاز به دروازه‌های دیگری از جمله XOR یاExclusive OR دارد، این دروازه‌ها توسط یک سلول قابل شبیه سازی نیستند اما ترکیبی از سلولها می‌توانند آنها را شبیه‌سازی کنند.



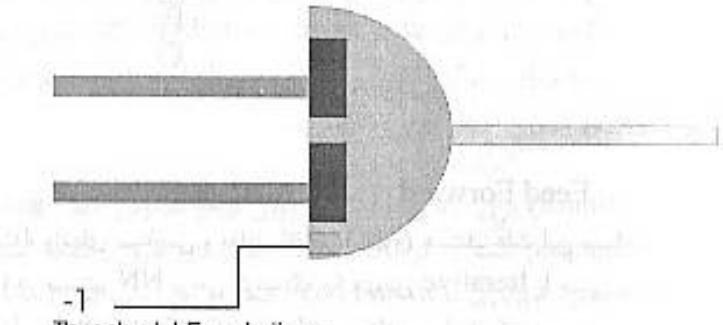
شکل ۷-۱: شباهت گراف خروجی Perceptron با عملیات منطقی OR

در عمل تابع ریا ضی ج سم سلولی نرون، تابعی کاملا دیجیتال با خروجی صفر یا یک نی ست، بلکه تابع آنالوگ شبیه به شکل ۸-۱ از این تابع می‌تواند تابع سیگموئید با یا تابع تانژانت هایپربولیک با شد و عملا ثابت شده که هر تابع غیر خطی دیگری می‌تواند کم یا بیش جایگزین آن شود از اینرو برنامه‌های شبیه سازی ANN معمولا انتخاب آن را به عهده کاربر می‌گذارند از این تابع با نامهای Active Function, Step Function, Nodal Function نیز یاد می‌شود.



شکل ۸-۱: تابع سیگموئید، تابع نمونه خروجی Perceptron

برای اینکه کار کردن با عدد آ ستانه یا Threshold آ سان با شد عمل گذاری (Implementation) آن طوری انجام می شود که شبیه یک اتصال سیناپسی مهاری (ودر مواقع خاص تحریکی عمل کند) به این منظور مطابق شکل ۹-۱ این مسئله به شکل یک اتصال سیناپسی مخفی با مقدار تحریک ثابت برابر با 1- (منفی یک) و وزن قابل تغییر شبیه سازی می شود با زیاد شدن وزن مربوطه، از آنجائیکه این وزن در منهای یک ضرب می شود مقدار آستانه تحریک سلول عصبی بالاتر می‌رود و تحریک بزرگتری لازم است تا سلول به مرحله شکلیک برسد.

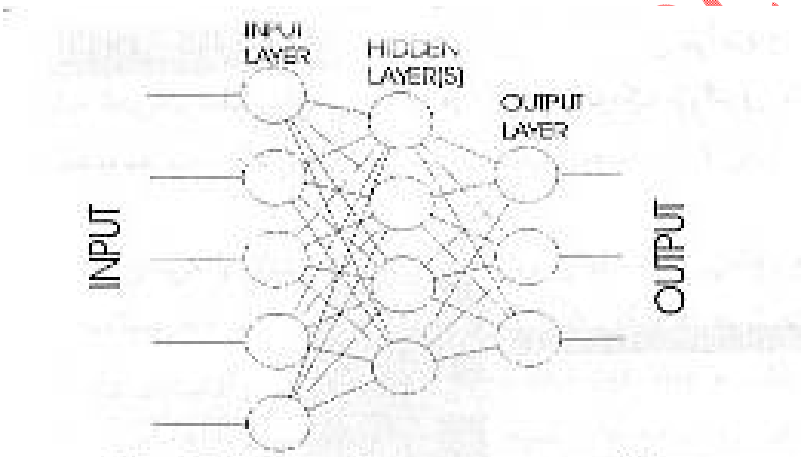


شکل ۹-۱: نحوه کارگزاری میزان آستانه یا Threshold

طرز کار شبکه عصبی:

از بهم پیو ستن سلول های مدل ع صبی م صنوعی(Artificial Neural Neural) بوجود می‌آید و ضعیت نسبی سلولها در شبکه (تعداد و گروه‌بندی و نوع اتصالات آنها) را توپولوژی شبکه می‌گو یند . توپولوژی در واقع سیستم اتصال سخت‌افزار نورونها به یکدیگر است که توام با نرم افزار مربوطه (یعنی روش ریاضی جریان اطلاعات و محاسبه وزنها)نوع عملکرد شبکه عصبی را تعیین می کنند. ساده‌ترین ومعمول‌ترین توپولوژی شبکه عصبی در شکل ۱۰-۱ نمایش داده شده است.

در این توپولوژی یک لایه ورودی وجود دارد که اطلاعات را دریافت می‌کند تعدادی لایه مخفی وجود دارند که اطلاعات را از لایه‌های قبلی می‌گیرند و به لایه‌های قبلی می‌دهند در نهایت یک لایه خروجی وجود دارد که نتیجه محاسبات به آنها می‌رود و جوابها در آن قرار می‌گیرند در هر سلول در هر لایه به کلیه سلولهای لایه مجاور بعدی متصل می‌شود اتصال به خود سلولها، به لایه قبلی، و پرش اتصالات در طول لایه ها مجاز نمی‌باشد. این توپولوژی به نام Feed forward معرف است زیرا جریان اطلاعات همیشه از ورودی به سوی خروجی است. بین ۹۰ تا ۹۵٪ کاربردهای شبکه های عصبی امروزی مربوط به این توپولوژی است.



شکل ۱۰-۱: توپولوژی Feed Forward

در ابتدا وزنهای سیناپسی، مقادیر اتفاقی (راندوم)هستند. علت این مسئله به روش محاسبه وزنها در سیستم NN برمی‌گردد که یک روش Iterative یا مبتنی بر تکرار است برای درک این مسئله فرض کنید که می‌خواهیم معادله زیر را حل کنیم:

$$X^4 + X^3 + 15.5X^2 + e^X + 2Sin(X) = 2.70$$

زمانی که سرعت محاسبه بالا است می‌توانیم یک راه حل کلی و ساده برای حل این معادله و هر معادله دیگر پیدا کنیم به طوری که هیچ کاری با حل جبری این معادله و معادلات دیگر نداشته باشیم کافی است یک مقدار دلخواه برای X فرض کنیم و آن را در معادله قرار دهیم و جواب معادله را حساب کنیم اگر تساوی برقرار بود مقدار فرض شده، پاسخ معادله است. در غیر اینصورت آنقدر آن را کم و زیاد می‌کنیم تا تساوی برقرار شود یا اینکه با خطای قابل قبولی به جواب نزدیک شویم. این روش حل معادله را روش مبتنی بر تکرار یا Iterative می‌گویند. همانطور که دیدیم، در این روش به یک مقدار حدسی اولیه نیاز داریم. این همان وزنهای اتفاقی سیناپسی هستند که در ابتدای کار با سیستم ANN استفاده می‌شوند.

در شروع مرحله یادگیری که به آن Learning یا Training هم گفته می‌شود، اطلاعات مربوط به نمونه‌ها یک یک به شبکه داده می‌شود. به طور مثال از طرفی غلظتهای شیمیایی و پارامترهای فیزیکی فرمولاسیون تک تک نمونه‌ها به لایه ورودی داده شده و از سوی دیگر مقدار تجربی و اندازه‌گیری شده پایداری آنها نیز به حافظه شبکه داده می‌شود (که مورد آخر به عنوان متغییری در حافظه کامپیوتر و خارج از توپولوژی شبکه نگهداری می‌شود) اطلاعات ورودی در شبکه جریان پیدا می کند به این معنی که در وزنهای سیناپسی ضرب شده و نتیجه فعالیت هر نورون خود به صورت سیگنالی، ورودی نورونهای لایه بعدی خواهد بود. در نهایت در پایان جریان اطلاعات هر نمونه، شبکه پاسخی در لایه خروجی خواهد داشت. مثلا اگر هدف پیشگویی پایداری فورمولاسیون است، وزن تنها نورون لایه خروجی نمایشگر آن خواهد بود، اگر چه در این مرحله ابتدایی، به دلیل استفاده از وزنهای سیناپسی اتفاقی، پاسخی ایجاد شده با میزان تجربی پایداری تفاوت خواهد داشت. این تفاوت (Prediction Error) که تفاضل میزان پیشگویی شده پاسخ توسط شبکه و میزان اندازه‌گیری شده تجربی آن است باید به صفر نزدیک شود تا شبکه از پیشگویی خوبی برخوردار شود.

معمول ترین روال (Routine) برای کاهش این خطا، روش توضیح معکوس خطا (Back propagation of Error) است که در 90 تا 95% کاربردهای امروزی شبکه عصبی روش مورد استفاده به همراه توپولوژی Feed Forward است در این روش، پس از محاسبه خطای پیشگویی، وزنهای سیناپسی از آخرین لایه به سوی نخستین لایه، به تدریج طوری تغییر می‌کند که خطای پیشگویی کمتر شود درواقع Back propagation سرشکن کردن خطا بر روی سلولهای لایه‌های مختلف است. پس از این، اطلاعات نمونه دوم به شبکه خورانده می‌شود. مسلما، با همان وزنه‌های سیناپسی، نمونه جدید مجددا خطا خواهد داشت. بنابراین روش توزیع معکوس مجددا د ست به کار شده و وزنها طوری تغییر می‌دهد که کمترین خطا را (هم برای این نمونه و هم برای نمونه پیشین) ایجاد کنند. به این ترتیب پس از خواندن تعداد کافی نمونه به ورودی شبکه، تمام فضای n بعدی روابط پارامترها توسط شبکه تجزیه و تحلیل می‌شود در این حالت گفته می‌شود که شبکه Converge شده ا ست به این معنی که در منحنی خطای پیشگویی به مقعرترین نقطه (minimum) ر سیده ا ست. به این معنای موفقیت در مرحله یادگیری ا ست و شبکه Converge شده آماده ا ست تا برای پیشگویی به کار رود.

اگر همه شرایط ایده آل باشد حال نوبت استفاده از شبکه است. با وارد کردن اطلاعات مربوط به یک نمونه جدید می‌توانیم میزان پایداری آنرا پیشگویی کنید. این در حالی است که هیچگونه فرضیه‌ای درباره نحوه برهمکنش و روابط میان پارامترهای ورودی نداشتید و هیچگونه محاسبه ریاضی نیز انجام ندادید این فوق العاده است!

توپولوژی شبکه:

لازم ا ست که نکاتی چند درباره توپولوژی شبکه بدانیم. یکی از پارامترهایی که تو سط کاربر در کلیه برنامه‌های شبیه‌سازی شبکه‌های عصبی قابل کنترل است تعداد سلول‌های هر لایه و تعداد لایه‌ها است و هر سلول عصبی یک گره (Node) نیز گفته می‌شود.

تعداد سلولهای لایه ورودی برابر با تعداد پارامترهای ورودی است در عمل سعی بر این است که کلیه پارامترهایی که در پاسخ تاثیر دارند در نظر گرفته شوند، البته باید در نظر گرفت که اطلاعات بی‌استفاده ورودی کار شبکه را مشکلتر می‌کنند زیرا اگر چه شبکه عصبی به نویز (داده‌های دارای خطا یا بی‌ربط) مقاوم است اما در هر صورت اگر میزان نویز بیش از حد زیاد باشد ممکن است شبکه نتواند Converge شود.

تعداد گره‌ها در لایه خروجی بستگی به پیشگویی مورد نظر ما دارد اگر قرار است شبکه میزان پایداری دارو را پیشگویی کند پس در ازاء داده‌های هر نمونه در مرحله یادگیر یک مقدار عدد (میزان پایداری تجربی اندازه‌گیری شده) به عنوان پاسخ مورد انتظار به آن داده می شود. به این ترتیب یک سلول در لایه خروجی کافی خواهد بود که مجموعه کلیه تحریکات وارد بر آن سلول برابر همان عدد مورد نظر ما خواهد بود. (Sum of Weighted Inputs) اگر قرار است که شبکه سرطانی بودن یا نبودن یک فرد را پیشگویی کنند پس در ازاء داده‌های هر فرد، در مرحله یادگیری یک ستون حاوی صفر یا یک عدد به شبکه داده می‌شود. صفر به معنای فرد سالم و یک به معنای فرد بیمار (سرطانی) خواهد بود. به این ترتیب یک سلول در لایه خروجی خواهد بود. اگر قرار بر این است که شبکه گروه خونی فرد را پیشگویی کند به تعداد گروههای خونی مورد نظر گره خواهیم داشت که روشن شدن هر کدام نشانه تعلق فرد به یک گروه خونی خاص خواهد بود.

تعداد لایه‌ها و تعداد گره‌ها در هر لایه مخفی از پارامترهایی است که توسط کاربر قابل تنظیم است با بررسی‌های دانشمندان مشخص شده که سیستم ادارک بشر (اعم از سیستم مرکز مغز و لایه‌های عصبی گسترده در بدن) جمعا یک شبکه ۶ تا ۷ لایه را تشکیل می‌دهند. هر چه تعداد لایه‌ها بیشتر باشد سیستم قادر به درک پیچیدگیهای بیشتر است. از لحاظ ریاضی اصطلاحا گفته می‌شود که افزودن هر لایه یک مرحله به قدرت Decode/ Encode اطلاعات می‌افزاید. بیشتر مسائلی که محققین با آن سر و کار دارند با دو تا سه لایه مخفی قابل حل هستند. وجود لایه‌های اضافی دقت عددی پیشگویی را کم می‌کند و ممکن است مانع Converge شدن شبکه گردد. به همین ترتیب کم بودن تعداد لایه‌ها نیز مانند کشف پیچیدگیهای روابط داده ها توسط شبکه می‌شود.

نکته دیگر تعداد گره‌ها در هر لایه است به طوری که تخمینی تعداد مناسب گره‌های لایه‌های مخفی بین نصف تا یک و نیم برابر تعداد گره‌های لایه‌ورودی (یعنی تعداد ورودیهای سیستم) است. منابع مختلف مقیاسهای تجربی متفاوتی به عنوان حدس اولیه برای این تعداد پیشنهاد کرده‌اند. در قسمتهای بعدی خواهیم دید که کم بودن مفرط تعداد گره‌ها قدرت تجزیه و تحلیل(و به دنبال آن دقت عدد پیشگویی) را کاهش می‌دهد.از سویی زیاد بودن مفرط این تعداد منجر می‌شود که سیستم بجای تجزیه و تحلیل داده‌ها آنها را حفظ کند (Memorizing) که این مسئله بسیار مهم در قسمتهای آتی بیشتر بحث خواهد شد.

ساختن یک مدل موفق به کمک شبکه‌های عصبی تا حدود زیادی منوط به تنظیم درست پارامترهای شبکه است تعداد این پارامترها (اعم از تعداد لایه‌های مخفی و تعداد گره‌های هر کدام، و نیز پارامترهایی که در جلوتر درباره آنها بحث خواهیم کرد) گاهی برای محققین مشکل ساز می‌شوند از اینرو امروزه روشهایی (یا برنامه‌هایی با کمک متدهای الگوریتم بهینه‌سازی این پارامترهای کمک می‌کنند. نکته قابل توجه اینکه برای داشتن یک شبکه که بتواند مدل مورد نظر ما را بسازد لازم نیست که پارامترهای نامبرده مقدار دقیق و خاصی داشته‌با شند بلکه قرار گرفتن هر یک از آنها در یک محدود (Range) خاص معمولا کافی است همچنین معمولا مقادیر پیشنهادی

و حد سی اولیه برای این پارامترها موجود ا ست. این م سئله باعث می شود که برای بی شتر م سائل روی آزمون و خطا بهینه سازی این پارامترها کافی به نظر برسد.

پارامترهای موثر در مرحله یادگیری:

در مرحله یادگیری در شبکه عصبی چند پارامتر ساده و مهم وجود دارند که در برنامه‌های مختلف کاربر می‌تواند آنها را دستکاری کند و یافتن بهینه مقدار آنها برای داشتن شبکه موفق و کارا لازم است. در اینجا به بحث مختصر درباره این پارامترها می‌پردازیم.

سرعت یادگیری (Learning Rate):

این اتصال سیناپسی واحد بین دو نورون را در نظر بگیرید فرض کنید که روال توزیع معکوس دستور العملی صادر کرده که وزن این اتصال از میزان W1باید به W2 تغییر یابد. تغییر وزن W1- W2 را گرادیان وزنی می‌نامند. این تغییر وزن برای کاهش خطای پیشگویی نمونه شماره ۲ لازم است اما **سبب تغییر خطای پیشگویی سایر نمونه‌ها می شود.** بسته به نوع سیستم، ما می‌توانیم تعیین کنیم که اهمیت هر د ستور العمل توزیع معکوس چقدر است یعنی سیستم در هر مرحله چقدر باید برای کاهش خطای پیشگویی عمل کند.

در سی ستم‌هایی که دقیق ه ستند و اعداد در آنها معنی ریا ضی و فیزیکی م شخ صی دارند به عبارتی دیگر در سیستم‌هایی که اهمیت هر نمونه زیاد است، می‌توانیم گرادیان را به طور تمام و کمال تاثیر بدهیم اما در سی ستم‌هایی که اعداد دارای خطای زیادی ه ستند اهمیت نمونه‌ها کم ا ست حتما باید تاثیر گرادیان را کاهش دهیم والا شبکه از هر نمونه به نمونه دیگر دچار تنش و تغییر شدید شده و هرگز Converge نمی‌شود. به طور مثال، اگر داده‌های ما مربوط به آزمایشات مربوط به عد سیها (فا صله، اندازه، و ماهیت حقیقی یا مجازی ت صویر) هستند، تاثیر گرادیان باید حداکثر باشد، اما اگر داده‌ها مربوط به پیشگویی ارزش سهام در بازار بورس یا پیشگویی موفقیت یک دانشجو در ارزشیابی‌های پایان‌تر هستند باید تاثیر این گرادیان کاهش یابد چرا که این بار سیستم یک سیستم دقیق ریاضی فیزیکی نیست و ارزش تک تک داده‌ها کمتر است.

به این منظور پارامتری به نام سرعت یادگیری یا Learning rate تعریف و در گرادیان ضرب می شود که گفته می شود متنا سب با اهمیت هر نمونه ا ست. اگر اهمیت هر نمونه زیاد با شد (م شابه اولین گروه سیستم‌ها) این پارامتر باید نزدیک به ۱ با شد اگر اهمیت هر نمونه کم با شد (مشابه د سته دوم سیستم‌ها) این پارامتر باید کمتر باشد.

Momentum:

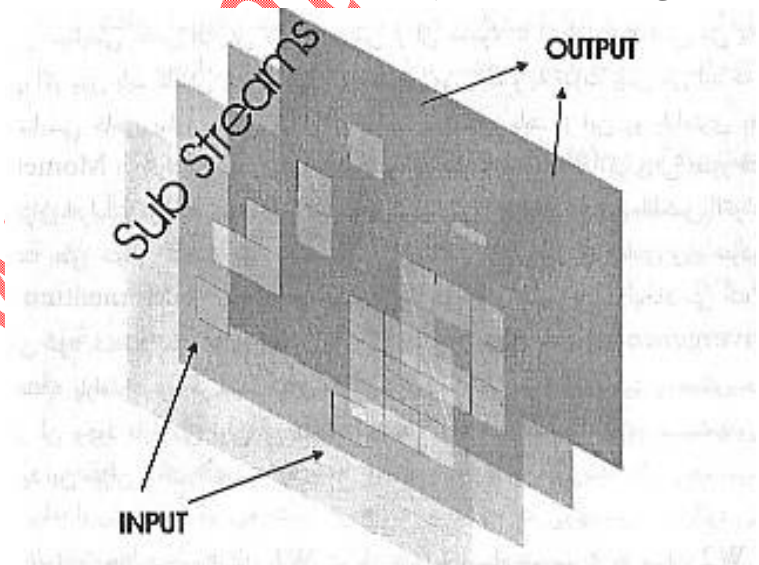
در شبیه‌سازی سیستم‌های مختلف مشاهده شده که معمولا زمانی که در طول فرایند یادگیری یک وزن سیناپسی خاص افزایش می‌یابد پس از آن معمولا تا انتها تنها افزایش می‌یابد و بلعکس. اگر وزن باید کاهش یابد تا انتها تنها کاهش می‌یابد و به ندرت پیش می‌آید که یک وزن سیناپسی خاص به طور متوالی کم و زیاد شوند و نواسان یابد از این رو پارامتری با نام **Momentum** در گرادیان تغییر وزن تاثیر داده می‌شوند به طوری که اگر وزنی به طور متوالی دستور برای افزایش دریافت کند به طور غیر عادی و زیادتری افزایش یابد و بلعکس، اگر وزنی قرار است به طور متوالی کاهش یابد به تدریج میزان کاهش آن به طور غیر عادی زیاد شود این پارامتر

Momentum نام دارد و اینر سی حرکتی برای تغییر وزنها ایجاد می‌کند که سبب می شود با تعداد نمونه‌های کمتر و در زمان کمتری سیستم به مرحله Convergence برسد. مشابه پارامتر قبلی در سیستم‌های دقیق که اهمیت هر نمونه زیاد ا ست و خطای عددی کمی در آن وجود دارد می‌توان از **Momentum** بالا ا ستفاده کرد اما در سیستم‌های غیر دقیق باید این میزان را کم انتخاب کرد.

در کل به ازاء هر وزن سیناپسی W1 که می‌باید به وسیله ضریب g به میزان W2 تغییر یابد مقدار g تحت تاثیر LR(میزان یادگیری) Mk (ضریب مومنتوم) قرار می‌گیرد که میزان Lr برای هر سیستم ثابت ا ست اما میزان Mk تحت تاثیر دو عامل است یکی میزان **Momentum** تعیین شده (که حداکثر Mk را تعیین می‌کند) و یکی سابقه تغییر وزن در همان گروه سیناپسی خاص (History یاHi) که اگر در چند مرحله متوالی در وزن سیناپسی مذکور تغییر مشابه (کاهشی یا افزایشی) مشاهده شود میزان Mk زیاد شده و به **Momentum** نزدیک می‌شود به این ترتیب، بر خلاف تاثیر ضریب یادگیری، تاثیر مومنتوم عملا بر روی اتصالات سیناپسی مختلف متفاوت است و تنها میزان حداکثر مجاز آن برای کل سیستم یکسان است. تساویهای زیر این نکات را یادآوری می‌کنند

$$mk = Momentum \times H$$
$$g = Lr \times mk$$
$$W_2 = g \times W_1$$

اگر بتوانیم تجسمی گرافیکی از یک شبکه Train شده (آموزش یافته)داشته باشیم چیزی مشابه شکل ۱۱-۱ خواهیم داشت که در آن تاثیرگذاری داده‌های ورودی از مسیرهای مربوط به وزنهای سیناپسی بالا " زیر جریانهایی" ارتباطی از درون لایه‌ها با خروجی ایجاد کرده است این تجسم کمک می‌کند که ما چنین جنبه مختلف در مورد شبکه عصبی را حس و درک کنیم.



شکل ۱۱-۱: تجسم جریان اطلاعات در یک شبکه آموزش یافته

اگر دقت کنیم می‌بینیم که عمل پردازش در مرحله کاربرد و پیشگویی یک شبکه عصبی آموزش یافته یک فرایند موازی ا ست چراکه کانالهای مختلف زیرجریانهای داده‌ها در کنار یکدیگر و به طور مستقل شروع شده و

تنها در آخر لایه لازم اســت که با هم جمع شــوند به همین دلیل با اســتفاده از پردازنده‌هایی که دارای قابلیت پردازش موازی ذاتی باشــند می‌توانند پیشــگویی بســیار ســریع انجام داد. در واقع به همین دلیل پردازنده های موازی مخصــوصــی در چند موســســه تحقیقاتی مختلف و متفاوت ســاخته شــده‌اند که به این منظور به کار برده می‌شوند.

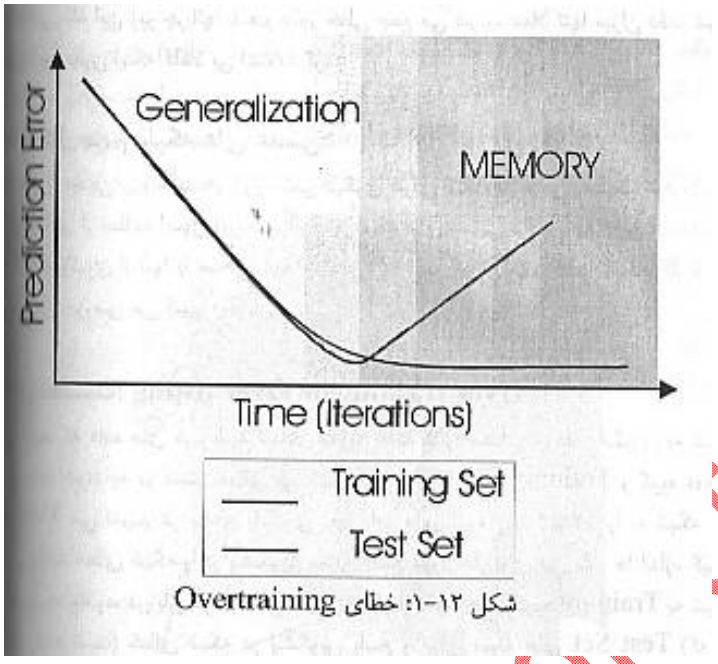
پردازش موازی قابلیت مهم دیگری به شبکه‌های عصبی می‌دهد و آن این که اگر بخشی از شبکه نابود شود یا اطلاعات از بین بروند تعدادی از زیرجریانهای مذکور از بین خواهد رفت و از آنجائیکه این زیرجریانها با هم به طور خطی جمع می‌شوند عملا تنها میزان دقت شبکه کم می‌شود بدون اینکه کاملا بی‌استفاده گردد.

سه اشکال مهم شبکه‌های عصبی :

شبکه های عصبی نیز مانند هر ابزار علمی دیگری دارای اشکالات خاص هستند هدف اصلی کاربران عادی از مطالعه اصــول زیربنایی طرز کار شــبکه‌های عصــبی درک ماهیت این اشــکالات و چگونگی جلوگیری از آنها یا تصحیح آنها است. در اینجا سه اشکال مهم که در هنگام کار با آنها پیش می‌آید بررسی می‌کنیم:

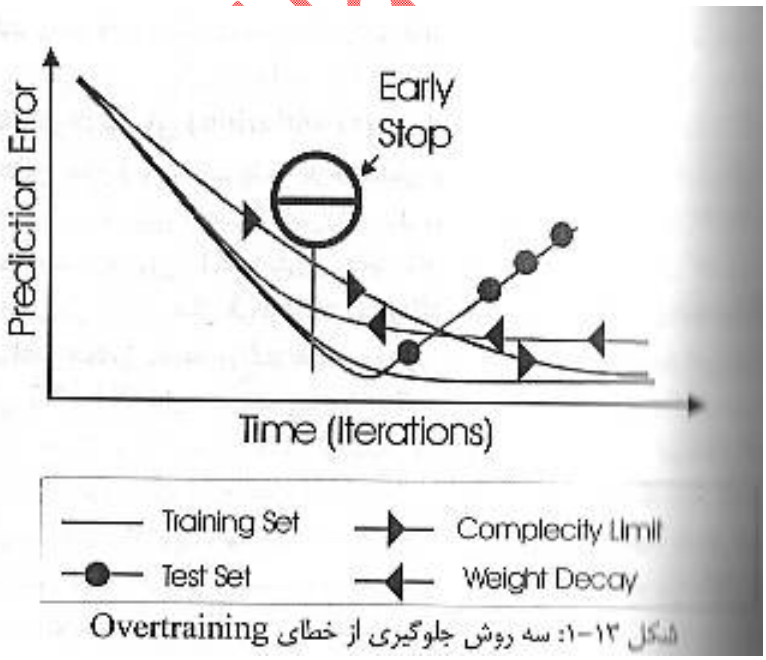
اشکال نخست: Over training or Over fitting

فرض کنیم که داده های مربوط به تعدادی نمونه را که قرار است در مرحله یادگیری به شبکه عصبی داده شود به دو دسته تقسیم می‌کنیم. گروه اول را Training Set و گروه دوم را Test Set می‌نامیم.در مرحله یادگیری تنها داده‌های گروه Training را به شکل می‌دهیم و دائما خطای شبکه را در پیشگویی میزان پاسخ مورد نظر برای این داده‌ها اندازه‌گیری می‌کنیم به علاوه در پایان هر مرحله (که یکی دیگر از نمونه‌ها از گروه Training به شبکه خورانده نشده و جدید هستند) نیز محاسبه می‌کنیم. همانطور که در شکل ۱۲-۱ نشان داده شده است مشــاهده می‌کنیم که پس از مدتی کاهش خطای کلی، شــبکه این قدرت را پیدا می‌کمد که با دقت بیشــتری نمونه‌های آشنا (Training) را پیشگویی کند اما خطایش در پیشگویی نمونه‌های جدید (Test)بیشتر و بیشتر می‌شود در واقع اتفاقی که در اینجا می‌افتد این است که شبکه به سوی سوق پیدا می‌کند که به جای تعمیم و قانون یابی (Generalization) بین نمونه‌ها سعی در حفظ کردن نمونه‌ها Memorization) می‌کند و به این دلیل از پیشگویی نمونه‌های جدید باز می‌ماند. در ست مانند دانش‌آموزی که به جای فهم مسائل فیزیک آنها را حفظ کند و آنگاه در موقع امتحان اگر به عین همان مسائل برخورد کند آنها را به خوبی حل می‌کند اما مسائل متفاوت را نمی‌تواند حل کند. این اتفاق راOverrtraining یا Over fitting می نامند. معمولترین اشکالی که کاربران شبکه‌های عصبی با آن مواجه هستند همین مسئله است. راههای متعددی برای جلوگیری از رسیدن به این حالت ناخواسته وجود دارد.



شکل ۱۲-۱: خطای

نخستین راه حل این است که در طول مرحله یادگیری ضریبی تعریف شود به طوری که اگر وزن نورونی ثابت بماند به تدریج آنرا فرسوده و کم کند، به عبارتی داده‌هایی که مورد استفاده نیستند دچار نوعی فراموشی می‌شوند به این روش Weight Decay گفته می‌شود. همانطور که در شکل ۱۳-۱ دیده می‌شود این روش مانع رسیدن شبکه به حداکثر دقت عددی ممکن می‌شود.



شکل ۱۳-۱: سه روش جلوگیری از خطای

روش دوم این است که تعداد گره‌های هر لایه شبکه را کم کنیم بطوریکه شبکه فضای کافی (حافظه کافی) برای بخاطر سپردن همه داده‌ها نداشته باشد و مجبور به تجزیه و تحلیل آنها شود. این روش را محدود کردن

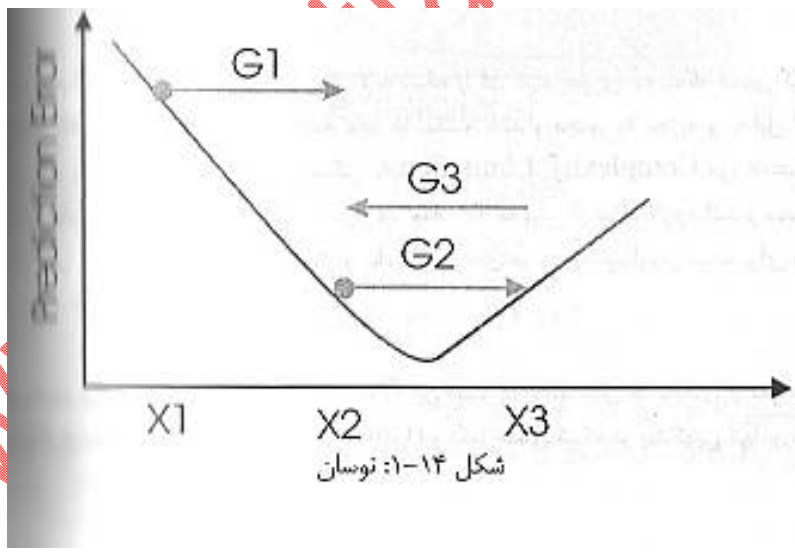
پیچیدگی یا Complexity Limitation می‌نامند. در طراحی اولیه توپولوژی شبکه باید دقت کرد که تعداد گره‌ها بیش از تعداد لازم نبا شد و معمولا مطالعه‌ای جهت بهینه سازی این پارامتر و سایر پارامترهای مربوط به توپولوژی شبکه برای حل هر مسئله خاص انجام می‌شود.

روش سوم برای جوگیری از Overtraining این است که مانند مثالها تعدادی از نمونه‌ها به دور از فرایند یادگیری بماند (Unexposed) و دایما خطای شبکه در پیشگویی آنها بررسی شود و هنگامی که این خطا رو به افزایش گذاشت فرایند یادگیری متوقف شود. این روش زود هنگام یا Early stop نامیده می‌شود.

در بیشتر برنامه‌های کامپیوتری موجود، از ترکیبی از روشهای دوم و سوم استفاده می‌شود .

اشکال دوم نوسان: (Oscillation)

این اشکال ناشی از این حالت بسیار ظریف ریاضی است که در کلیه روشهای مبتنی بر تکرار بوجود می‌آید. یک وزن سیناپسی خاص را در نظر بگیرید. همانطور که در شکل ۱۴-۱ نشان داده شده است، این وزن (X1) دستوری مطابق g1 برای تغییر وزن دریافت می کند و به وزن X2 تبدیل می‌شود. در شکل می‌بینیم که دستور g1 ماثر بوده و سبب کاهش خطا شده ا ست. اما در ادامه هنگامی که د ستور g2 صادر می شود و به وزن X3 تبدیل می‌شود به دلیل قرینگی X2 وX3 حول محور تقارن منحنی خطا، میزان خطا ثابت می ماند و کمتر نمی‌شود.در مرحله بعد اگر مجددا دستور مبتی بر افزایش وزن صادر شود خطا بیشتر می‌شود بنابراین دستور g3 به میزان g2 اما خلاف جهت آن صادر می‌شود. اگر باز هم g3 تکرار شود روی منحنی خطا بالا خواهیم رفت به طوریکه به X1 می‌ر سیم، پس باز باید جهت د ستور گرادیان بعدی عکس شود به این ترتیب سی ستم ما دائما دستور g2 وg3 را به تناوب دریافت خواهد کرد و بین دو نقطه X2 وX3 نوسان خواهد کرد.



شکل ۱۴-۱: نوسان

برای حل این مسئله باید میزان قدمها یا اندازه g را بتوان تغییر داد تا به بهترین نقطه (پائین‌ترین نقطه منحنی خطا) بر سیم. برای این امر یک کنترل کننده ریا ضی، میزان عددی g را طوری کم و زیاد می‌کند که سیستم از حالت نوسان خارج شود.

مموا این کنترل کننده بر اساس ریاضیات فازی (Fuzzy) طراحی می‌شود. این کنترل کننده معمولا در دسترس کاربر نمی‌باشد و به طور خودکار وظیفه دارد که میزان momentum را طوری تغییر بدهد که گرادیان g در کوتاهترین زمان سیستم را به بهترین نقطه هدایت کند بدون اینکه به نوسان بیافتد.

اشکال سوم: Co- Existence یا Co- Linearity

این اشـــکال را با یک مثال توضـــیح می‌دهیم. در دهه ۱۹۸۰ پنتاگون تصـــمیم گرفت تا از تکنولوژی کامپیوتر و شبکه‌های عصبی برای محافظت تانکهایش استفاده کند. برای این کار دوربینی دیجیتالی بایست دائما تصاویر محیط اطراف را گرفته و تجزیه و تحلیل کند. برای تجزیه و تحلیل تصاویر از یک شبکه عصبی استفاده شد به این ترتیب که در مرحله یادگیری ۱۰۰ تصـــویر از مناظری که در آنها تانکی پنهان شـــده بود و ۱۰۰ تصـــویر از مناظر بدون تانک گرفته شد و از هر گروه ۵۰ تصویر به شبکه عصبی خورانده شد. نتیجه بسیار خوب بود شبکه با دقت بسیار عالی می‌توانست از ۱۰۰ عکسی که کنار گذاشته شده بود تصاویر دارای تانک را جدا کند.

دانشـــمندان برای اطمینان بیشـــتر تعدادی عکس جدید تهیه کردند و این بار نتیجه‌ای متفاوت دریافت کردند: پاسخها کاملا اتفاقی بود و اثری از آن دقت اولیه نبود. پس از مدتی مطالعه آنها متوجه شدند که در تصاویر اولیه، در تمام تصاویری که تانک وجود نداشت آسمان کاملا آفتابی بود. به این ترتیب وجود تانک و عدم وجود آسمان آفتابی کاملا همزمان اتفاق می‌افتند و پروژه چند میلیون دلاری مذکور تنها این فایده را داشت آسمان آفتابی را از غیر آفتابی جدا می‌کرد.

این اشکال در واقع نه به طرز کار شبکه عصبی، بلکه به طراحی بد مسئله را یا ارائه بد داده‌های ورودی مربوط است. شبکه مذکور ساده‌ترین و مطمئن‌ترین راه حل برای جدا کردن دو گروه عکسها به درستی پیدا کرده بود. بنابراین در هنگام انتخاب اطلاعات ورودی در مرحله یادگیری باید دقت کرد. فرض کنید اطلاعات افراد را برای پیشگویی یک بیماری وارد سیستم می‌کنیم. اگر اطلاعات بیماران را از شماره ۱ تا ۵۰ وارد کنیم و اطلاعات افراد سالم را از ۵۰ تا ۱۰۰ در صورتی که شماره ترتیبی را هم به عنوان داده به شبکه وارد کنیم شبکه به سرعت ارتباط درست اما ناخواسته مربوطه را درک می‌کند و از این پس تنها و تنها بر اساس شماره ترتیبی نمونه‌های جدید آنها را به دو گروه بیمار و سالم تقسیم خواهد کرد و کاری با پارامترهای دیگری نخواهد داشت.

در این مثال نیز با دادن داده‌های چند نمونه کاملا جدید اشکال فوق مشخص می شود و با حذف ستون شماره بیمار از داده‌های ورودی و تکرار مرحله یادگیری این اشکال رفع خواهد شد.

فصل دوم:

پیاده‌سازی مدل شبکه عصبی مصنوعی برای دسته بندی بیماران کمخون

در این فصل قدم به قدم مراحل پیاده‌سازی یک سیستم ساده شبکه عصبی که بر اساس شاخصهای موجود در گزارش CBC، بیماران کمخون را دسته بندی می‌کند، خواهیم پرداخت.

مراحل پروژه:

مراحل پروژه را می‌توانیم در چهار قسمت در نظر بگیریم.

- طراحی شبکه عصبی
- تهیه داده‌هایی که قرار است شبکه عصبی مصنوعی بر اساس آن آموزش یابد.
- آموزش شبکه عصبی بر اساس داده‌های آماده شده در مرحله قبل.
- استفاده از شبکه عصبی با داده‌هایی که در آموزش آن نقش نداشته‌اند، که این مرحله هم صحت واقعی شبکه عصبی تعیین خواهد شد و هم روش استفاده از شبکه عصبی را مشخص می‌کند.

طراحی شبکه عصبی:

ابتدا باید ورودیها و خروجیها و ساختار شبکه عصبی مصنوعی را مشخص نماییم. در این پروژه ورودیهای شبکه عصبی مصنوعی عبارت خواهند بود از: هموگلوبین (HGB)، MCV و RDW. که همگی در یک گزارش شمارش سلولهای خونی معمولی موجود می‌باشد. همچنین برای سادگی موضوع خروجی را فقط به صورت سه دسته سالم، فقر آهن و تالاسمی مینور در نظر خواهیم گرفت.

شبکه عصبی ما از دولایه تشکیل خواهد شد. لایه مخفی (Hidden) و لایه خروجی.

تهیه داده‌ها جهت آموزش شبکه عصبی:

برای آموزش شبکه عصبی به اطلاعات بیماران واقعی نیاز داریم. این اطلاعات به ازای هر بیمار باید شامل HHB، MCV، RDW و همچنین تشخیص اینکه آیا فرد سالم یا کمخون با فقر آهن یا تالاسمی مینور است. به عنوان مثال باید اطلاعات فوق را برای ۲۰۰ فرد به شکل دو ماتریس داشته باشیم. ماتریس اول ورودی شکبه خواهد بود و شامل ۳ شاخص فوق می‌باشد. ماتریس دوم تشخیص این افراد را به ترتیب شامل می‌شود.

در شرایط واقعی اطلاعات افراد را باید از مواردی که نتیجه آزمایش CBC و تشخیص نهایی آنها در دست است، جمع آوری نمود. به عنوان مثال می‌توان این کار را در یک بیمارستان انجام داد و مانند یک تحقیق روتین پزشکی به بررسی پرونده‌های موجود در آن بیمارستان پرداخت.

از آنجا که جمع‌آوری اطلاعات تعداد زیادی بیمار کاری است وقت‌گیر و در حوصله این نوشته نیست، برای تهیه این اطلاعات نیز یک مدل ریاضی در برنامه MATLAB پیاده سازی می‌نماییم. مدلهای ریاضی ذکر

شده در مقدمه همه به صورت خطی هستند و موضوع را بسیار ساده نموده‌اند و مدل نمودن مجدد آنها با شبکه عصبی، کار جالبی نیست و شبکه به سرعت به نتیجه می‌رسد که البته مزیتی محسوب نمی شود. بنابراین با استفاده از میانگین و انحراف معیاریهای ذکر شده در مقالات برای شاخصهای سه‌گانه خونی مورد نظر ما، سعی خواهیم کرد، مدل ریاضی پیچیده‌تری را باز سازی کنیم.

مدل ریاضی فوق را به شکل $y=f(x)$ در نظر بگیرید. y یا خروجی ما باید دو ماتریس ورودی و خروجی شبکه عصبی را تامین نماید. x نیز تعداد فردی است که ما از مدل می‌خواهیم که اطلاعات آنها را برای ما شبیه سازی کند. پس باید یک تابع در برنامه MATLAB به صورت زیر بسازیم:

```
[P,T]=AnemiaGetData(c)
```

c تعداد افراد مورد نیاز ما ست. P ماتریس c*۳ ا ست که شاخصهای خونی هر بیمار را در هر ستون خود دارد و T نیز یک ماتریس c*۳ است که تشخیص را که به صورت تصادفی بدست آمده است، در خود دارد و به این شکل است که هر ستون به ازای یک بیمار است و بر اساس تشخیص آن بیمار یکی از ردیفهای آن عدد ۱ و دو ردیف دیگر عدد ۰ را در خود دارد.

کد این تابع در فایل AnemiaGetData.m قرار دارد.

حال به توضیح خط به خط کد فایل فوق می‌پردازیم:

```
function [P,T]=AnemiaGetData(c)
```

خط اول، ورودی و خروجی تابع را مشخص می‌کند.

```
% Version 1.1.1
% c: Samples count
```

شماره نسخه را از این جهت در این قسمت ثبت می‌کنیم تا در صورت استفاده از نتایج این تابع در یک شبکه بتوان شماره نسخه در آن شبکه آموزش دیده، ثبت شود و با تغییر پارامترهای داخلی این تابع، شماره نسخه را نیز تغییر داد تا مشخص شود که شبکه برای این داده‌ها آموزش دیده است یا خیر.

```
d=fix(rand(1,c) * 3)+1;
```

عبارت فوق یک ماتریس c*۱ را ایجاد می‌کند که اعضای آن اعداد تصادفی ۱ تا ۳ می‌باشند.

```
P=zeros(3,c);
T=zeros(3,c);
```

دو خط فوق دو ماتریس c*۳ که همه اعضای آن عدد صفر هستند را ایجاد می‌کند.

حال باید مقادیر اعضای دو ماتریس فوق را به صورت مورد نظر تغییر دهیم:

```
for i=1:c
```

دستورهای switch و case بر اساس هر یک از مقادیر ماتریس d که در حقیقت نشان دهنده یکی از تشخیصهای سه‌گانه است، کد خاصی را اجرا می‌کنند و به این ترتیب مقادیر ماتریسهای T و P را تنظیم می‌کنیم.

```
    switch d(1,i)
```

```
case 1 %Normal
```

در حالتی که تشخیص عدد ۱ باشد یعنی بیمار سالم است.

```
T(1:3,i)=[1;0;0];
```

در این حالت ستون مربوطه در ماتریس T به این شکل ا ست که ردیف اول عدد ۱ و ردیفهای دیگر عدد ۰ در خود دارند.

توجه: برای بد ست آوردن اعداد مورد نظر چنان که می‌بینید از تابعی به نام myrandn ا ستفاده شده ا ست. این تابع پس از انتهای تابع فعلی تعریف شده ا ست. در این مرحله پی‌شنهاد می شود که ابتدا بخش تو ضیح این تابع که در چند صفحه بعد قرار دارد، مطالعه شود.

در ماتریس P هر ستون شامل سه عدد ا ست که عدد اول ن شان دهنده میزان هموگلوبین، عدد دوم میزان MCV و عدد سوم میزان RDW است.

```
%HB
P(1,i)=myrandn(14,1); %12-16
```

در کد فوق می‌خواهیم با فرض سالم بودن بیمار، میزان هموگلوبین او را تعیین کنیم که برای این کار عدد تصادفی با توزیع نرمالی را ایجاد می‌کنیم که میانگین آن عدد ۱۴ و انحراف معیار آن عدد ۱ است. بنابراین به احتمال ۹۵٪ این عدد بین ۱۲ تا ۱۶ است که همان اعداد هموگلوبین در یک فرد (مرد) سالم است.

توجه: در بدست آوردن شاخصهای سه‌گانه خونی جهت حفظ سادگی، جنسیت در نظر گرفته نشده است.

```
%MCV
P(2,i)=myrandn(90,5); %80-100
```

به همین منوال کد فوق مقادیر MCV در یک فرد سالم را ایجاد می‌نماید.

```
%RDW
P(3,i)=myrandn(13,0.7); %11.6-14.4
```

و همچنین مقادیر RDW.

```
case 2 %Iron Deficiency
```

در کد زیر نیز مقادیر شاخصهای سه‌گانه برای یک فرد با فقر آهن تولید می‌شود.

```
T(1:3,i)=[0;1;0];
%HB
P(1,i)= myrandn(9,2); %5-13
%MCV
P(2,i)=myrandn(60 + (P(1,i) - 6 ) * 4 ,2); %52-92
```

از آنجا که هر چه فقر آهن شدیدتر باشد هموگلوبین مقادیر پایین‌تری میابد و اندازه گلوبولها و مقدار MCV نیز کوچکتر می‌شود، در کد فوق مقدار MCV به مقدار هموگلوبین وابسته شده است تا هماهنگی بیشتری با هم داشته باشند.

```
%RDW
P(3,i)=myrandn(16,1); %14-18
```

```
        case 3 %Thalasemia
```
در کد زیر نیز مقادیر شاخصهای سه‌گانه برای یک فرد با تالاسیمی مینور تولید می‌شود.
```
            T(1:3,i)=[0;0;1];
            %HB
            P(1,i)= myrandn(12,1); %10-14
            %MCV
            P(2,i)=myrandn(64,8); %48-80
            %RDW
            P(3,i)=myrandn(19,1.5); %16-22
    end
  end

end
```

در این قسمت کد تابع به پایان می‌رسد. به این ترتیب با لوپی که با استفاده از دستور for ایجاد شده است، هر دو ماتریس P و T مقادیر مورد نظر را پیدا می‌کنند. به این ترتیب ستون اول ماتریس P شاخصهای سه‌گانه خونی را در خود دارد و ستون اول ماتریس T تشخیص بیمار اول و به همین ترتیب ستونهای دیگر این دو ماتریس، اطلاعات بیماران دیگر را در خود دارند.

تابع myrandn :

همانطور که قبلا هم ذکر شد تابع AnemiaGetData مقادیر هر یک از شاخصهای سه‌گانه خونی مورد استفاده برای ورودی شبکه عصبی را بر اساس میانگین و انحراف معیار ذکر شده در مقالات بر اساس تشخیصهای بیماری، بدست می‌آورد. بنابراین باید تابعی داشته باشیم که بر اساس میانگین و انحراف معیار داده شده یک عدد تصادفی با توزیع نرمال، ایجاد کند. تابع randn که از توابع پایه MATLAB است، یک عدد تصادفی را با فرض نرمال بودن توزیع و ۰ بودن میانگین و ۱ بودن انحراف معیار ایجاد می‌کند. ولی آنچه ما در این مرحله به آن نیاز داریم این است که تابعی شبیه randn داشته باشیم با این تفاوت که میانگین و انحراف معیار آن را خودمان مشخص کنیم.

حال به بررسی کد زیر که به هدف فوق نوشته شده است، می‌پردازیم:

```
function [r]=myrandn(mean,std)
```

در کد فوق mean میانگین و std انحراف معیار مورد نظر برای ایجاد عدد تصادفی با فرض نرمال بودن توزیع است. r عدد تصادفی ایجاد شده است.

```
   r= mean + randn * std;
```

کد فوق هدف ما را برآورده می‌سازد عدد r یک عدد تصادفی با توزیع نرمال با مانگین mean و انحراف معیار std خواهد بود.

```
   r=fix(r * 100)/100;
```

کد فوق عدد r را تا ۲ رقم اعشار گرد می‌کند.

end

در این قسمت کد مربوط به تابع myrandn به پایان می‌رسد.

ایجاد شبکه عصبی:

جعبه ابزار (Toolbox) شبکه عصبی برنامه MATLAB کار *ایجاد* مدل شبکه عصبی را بسیار آسان ساخته است. در این جعبه‌ابزار، فهرستی از توابعی که برای اهداف مختلف شبکه‌های عصبی مختلفی را *ایجاد* می‌نمایند، وجود دارد. در زیر فهرست این توابع را می‌بینند:

network	Create a custom neural network.
newc	Create a competitive layer.
newcf	Create a cascade-forward backpropagation network.
newelm	Create an Elman backpropagation network.
newff	Create a feed-forward backpropagation network.
newfftd	Create a feed-forward input-delay backprop network.
newgrnn	Design a generalized regression neural network.
newhop	Create a Hopfield recurrent network.
newlin	Create a linear layer.
newlind	Design a linear layer.
newlvq	Create a learning vector quantization network.
newp	Create a perceptron.
newpnn	Design a probabilistic neural network.
newrb	Design a radial basis network.
newrbe	Design an exact radial basis network.
newsom	Create a self-organizing map.

جزییات هر یک از این توابع و هدف هر یک آنها خود بحثی کامل می‌خواهد و نیاز به اطلاع از جزییات ریاضی و منطق پشت هر یک از روشهای مختلف شبکه‌های عصبی دارد که از حوصله این نوشته خارج است. در این بحث، ما از تابع newlvq استفاده می‌کنیم که از جمله دلایل استفاده از این تابع، کیفی بودن خروجی شبکه ماست.

کد زیر تنها خط لازم برای ایجاد شبکه عصبی در مثال ماست:

```
net = newlvq(minmax(P),10,[.3 .3 .4]);
```

net شبکه ایجاد شده است.

P ماتریس ورودی شبکه است که ما آن را توسط تابع AnemiaGetData ایجاد خواهیم نمود.

تابع minmax مقادیر حداقل و حداکثر موجود در ماتریس P را برمی‌گرداند.

عدد ۱۰ نشان دهنده تعداد نرون لایه مخفی است.

[.4 .3 .3] ماتریسی است که نسب تخمینی دسته‌های خروجی را مشخص می‌کند.

همانطور که گفته شد با یک خط کد فوق، شبکه عصبی مورد نظر ما ایجاد شده است. جزییات بیشتر شبکه را می‌توان در مرحله آموزش تعیین نمود.

آموزش شبکه عصبی:

بعد از ایجاد شبکه و فراهم بودن اطلاعات لازم برای آموزش شبکه با آن، می‌توان کار آموزش شبکه را شروع نمود. درک جزییات آموزش شبکه نیازمند اطلاع از مفاهیم ریاضی آن است که از آنجا که هدف این نوشته آشنایی پزشکان محقق با کاربردهای شبکه عصبی و نحوه استفاده از برنامه MATLAB است، بنابراین مفاهیم ریاضی آن از حوصله این بحث خارج است. البته برنامه MATLAB این مساله را نیز بسیار آسان نموده است و شما می‌توانید بدون درگیر شدن با مفاهیم ریاضی آن، به هدف خود در استفاده از شبکه عصبی دست یابید.

حال به بررسی کد آموزش شبکه‌ای که در بالا بحث نمودیم، می‌پردازیم:

کد ایجاد و آموزش شبکه عصبی در فایل AnemiaTrain.m قرار دارد.

```
function AnemiaTrain()
  clc;
```

کد فوق صفحه برنامه MATLAB را پاک می‌نماید.

```
  [P,T]=AnemiaGetData(50);
```

کد فوق با استفاده از تابع AnemiaGetData که قبلا توضیح داده شده است، ماتریس P به عنوان ورودی شبکه و ماتریس T را به عنوان خروجی یا هدفی که شبکه باید به آن برسد را ایجاد می‌نماید. چنانچه می‌بینید این ماتریسها برای ۵۰ بیمار فرضی مشخص شده است.

```
  net = newlvq(minmax(P),10,[.3 .3 .4]);
```

کد فوق که قبلا توضیح داده شده است، شبکه را ایجاد می‌کند.

```
  net.trainParam.show = 5;
```

کد فوق تعیین می‌نماید که هر ۵ بار آموزش، گراف آموزش که در ذیل توضیح داده خواهد شد، تازه‌سازی شود.

```
  net.trainParam.lr = 0.005;
```

کد فوق Learning Rate را که قبلا توضیح داده شده است، مشخص می‌سازد.

```
  net.trainParam.mc = 0.9;
```

کد فوق مقدار Momentum را که قبلا توضیح داده شده است، مشخص می‌نماید.

```
  net.trainParam.epochs = 3000;
```

کد فوق حداکثر تعداد epoch را مشخص می‌نماید. epoch به هر بار آزمایش اطلاعات همه نمونه‌ها و تعیین وزنهای شبکه بر اساس آنها گفته می‌شود.

```
  net.trainParam.goal = 1e-3;
```

کد فوق حداکثر میزان خطای مورد نظر را مشخص می‌کند.

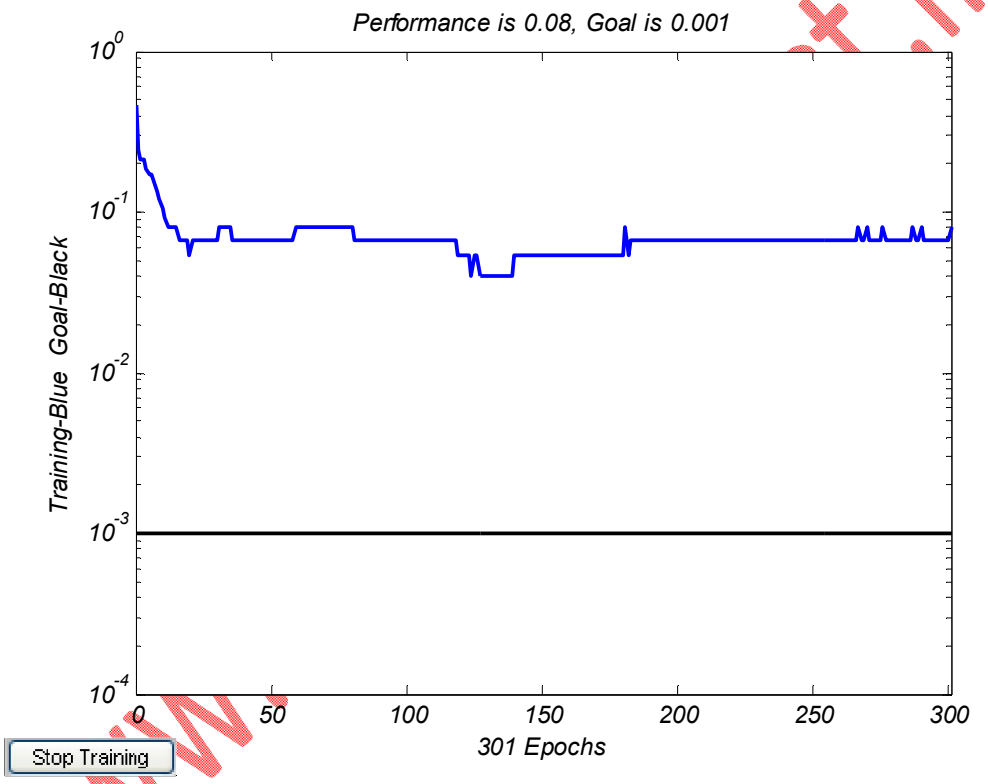
```
net.trainParam.time = 1000;
```

کد فوق حداکثر زمان آموزش را بر حسب ثانیه مشخص می‌کند.

```
[net]=train(net,P,T);
```

کد فوق آموزش شبکه را آغاز می‌نماید و نتیجه را که شبکه‌ای است که وزنهای آن تغییر کرده است، مجددا در net می‌ریزد. با اجرای این کد فرمی که گراف آموزش را نشـــان می‌دهد، باز می‌شـــود و میزان خطا را بر حسب epoch نشان می‌دهد.

در زیر تصویری از یک گراف آموزش دیده می‌شود:



خط سیاهی که در 10^{-3} دیده می‌شود نشان دهنده خطایی است که ما می‌خواهیم به آن برسیم.

با اجرای د ستور train کار آموزش شبکه آغاز می شود و تا زمانی ادامه می‌یابد که یا به میزان خطای تعیین شده بر سد یا با یکی از محدویهای تعداد epoch یا زمان، یا توقف د ستی تو سط کاربر، کار آموزش متوقف شود.

اگر شـــبکه نتوانســـت در محدودیتهای تعیین شـــده به خطای مورد نظر برســـد می‌توان کار آموزش را مجددا تکرار نمود. از آنجا که تصادف در وزنهایی که در حین آموزش به شبکه داده می شود دخیل ا ست، تکرار کار آموزش می‌تواند نتایج متفاوتی را در برداشـــته باشـــد. همچنین تعداد بیمار فرضـــی، تعداد نرون لایه مخفی، Learning Rate و Momentum را نیز می‌توانید تغییر دهید و نتایج متفاوتی به دست آورید.

فایل Net001(1-1-1).mat حاوی شبکه‌ای است که با پارامترهای گفته شده در بالا آموزش دیده است و به خطای حین آموزش 0.001 رسیده است. همچنین شماره نسخه پارامترهای داخلی تابع AnemiaGetData نیز 1.1.1 بوده است.

در این مرحله اجرای کد وارد قسمتی می‌شود که هدف آن ذخیره شبکه آموزش دیده در یک فایل است. چهار خط زیر این وظیفه را بر عهده دارند تا بتوان بعدا از این شبکه آموزش دیده، استفاده نمود.
این کد باعث می شود انتهای کار از کاربر سئوال "'What is file name to save this net?" پرسیده شود. در صورتی که متنی جلوی آن تایپ شود، فایل به همان نام و با پسوند mat ذخیره خواهد شد. در صورتی که نمی‌خواهید فایل ذخیره شود، فقط دکمه Enter را بزنید.

```
R = input('What is file name to save this net?','s');
if ~isempty(R)
   save(R,'net')
end

end
```

در این مرحله کار کدینگ آموزش به پایان می‌رسد.

بررسی میزان خطای شبکه عصبی آموزش دیده:
پس از اینکه کار آموزش به پایان رسید، حال باید آن را با اطلاعاتی که در آموزش مورد استفاده قرار نگرفته‌اند، مورد آزمایش قرار داد و میزان خطای واقعی آن را تعیین نمود.

کد تعیین میزان خطای شبکه آموزش دیده در فایل AnemiaSim.m قرار دارد.

```
function AnemiaSim(netfile)
   clc;
```

netfile اسم فایلی است که شبکه آموزش دیده در آن ذخیره شده است.

```
   load(netfile);
```

کد فوق شبکه ذخیره شده در فایل را در حافظه بازگذاری می‌کند.

```
   n=1000;
   [P,T]=AnemiaGetData(n);
```

کدهای فوق اطلاعات ۱۰۰۰ نفر فرضی که می‌خواهیم اطلاعات آنها را برای بررسی شبکه مورد استفاده قرار دهیم، تامین می‌نماید.

```
   a=sim(net,P);
```

دستور sim شبکه و اطلاعات ورودی را تحویل گرفته و آنها را بر اساس تشخیصهای سه‌گانه دسته‌بندی می‌نماید. پس اگر ما اطلاعات یک بیمار را در یک ستون ماتریس P قرار دهیم، خروجی a تعیین کننده

تشخیص شبکه عصبی برای این بیمار است. همچنین اگر ما a بدست آمده برای هزار بیمار فوق را با T مقایسه نماییم میزان خطای شبکه بدست می‌آید.

حال در کدهای زیر می‌خواهیم نسبت موارد تشخیص درست شبکه به کل موارد بررسی شده را بدست آوریم:

```
e=0;
for i=1:n
   if T(1:3,i)==a(1:3,i)
      e=e+1;
   end
end

per=e/n;
per
```

عدد per نسبت موارد تشخیص درست شبکه به کل موارد بررسی شده را نشان می‌دهد.

```
end
```

حال می‌توانید با استفاده از شبکه آموزش دیده‌ای که در فایل Net001(1-1-1).mat ذخیره شده است، کد فوق را آزمایش نمایید.

برای این کار در برنامه MATLAB دستور زیر را اجرا نمایید:

```
>>AnemiaSim('Net001(1-1-1)')
```

که خروجی شبیه زیر را دریافت خواهید کرد که البته در تکرار کد فوق عددهای متفاوتی را تجربه خواهید نمود.

```
per =
   0.9010
```

این عدد به این معناست که این سیستم در مورد افراد فرضی که اطلاعات آنها در آموزش به کار نرفته است، می‌تواند با استفاده از شاخصهای خونی سه‌گانه، سلامت یا نوع بیماری فرد را با صحت بیش از ۹۰ در صد تشخیص دهد.

نتیجه‌گیری:

حال فرض کنید که شبکه فوق را بر اساس اطلاعات افراد واقعی آموزش داده بودیم و به میزان خطای حین آموزش قابل قبولی رسیده بودیم و سپس با استفاده از اطلاعات افراد واقعی دیگری که در آموزش شبکه از آنها استفاده نشده بود، دقت واقعی شبکه را سنجیده بودیم و شبکه توانسته بود با صحتی بالای ۹۵ درصد برای این افراد تشخیص درست بگذارد، ما به سیستمی دست یافته بودیم که اگر آن را به دستگاه CBC خودکار که یک رایانه داخلی دارد، اضافه می‌نمودیم، می‌توانستیم در گزارشهای CBC علاوه بر شاخصهای

روتین خونی، تشخیص علت کمخونی را نیز با صحت قابل قبولی داشته باشیم و پزشک مجبور نبود برای کشف علت MCV پایین آزمایشهای پیچیده‌تری را درخواست نماید.

منابع:

1. Birndorf NI, Pentecost JO, Coakley JR, Spackman KA, An expert system to diagnosis anemia and report results directory on hematology forms, Comput Biomed Res. 1996 Feb;29(1):16-26.
2. Y. AKAI, F. KUBOTA, A Study of b Thalassemia Screening using an Automated Hematology Analyzer, Sysmex J Int 8 : 110-114, 1998
3. Dorland Dictionary, Edition: 28
4. MATLAB Help, Version: 7.0
۵. تجزیه و تحلیل داده‌ها به وسیله شبکه‌های عصبی مصنوعی و کاربرد آن در علوم پزشکی، هومن شادنیا، مرکز ملی تحقیقات علوم پزشکی کشور، چاپ اول، بهار ۸۳