به نام خدا

Kohonen Network

(Self Organization Map)

گردآوری

مهدی صالح پور، امیر حبیب زاده

زیر نظر استاد

آقای دکتر هادی صدوقی یزدی

(دانشگاه تربیت معلم سبزوار)

کلمات کلیدی

شبکه های عصبی، *Kohonen*.

چکیده

شبکه عصبی مصنوعی یا *ANN (Artificial Neural Network)* یک نمونه سیستم پردازش است که درآن از سیستم های عصبی بیولوژیکی مانند مغز الهام گرفته شده است . عضو کلیدی این ساختار جدید سیستم پردازندهٔ اطلاعات می باشد که تعداد زیادی از آنها به صورت مجتمع مانند هورمونهای مغز با یکدیگر کار می کنند تا بتوانند مسائل خاصی مانند تشخیص الگو یا طبقه بندی داده ها را از طریق فرایند یادگیری حل نمایند .

۱.شبکه عصبی چیست ؟

شبکه عصبی مصنوعی یا *Artificial Neural Network (ANN*) یک نمونه سیستم پردازش است که درآن از سیستم های عصبی بیولوژیکی مانند مغز الهام گرفته شده است . عضو کلیدی این ساختار جدید سیستم پردازندۀ اطلاعات می باشد که تعداد زیادی از آنها به صورت مجتمع مانند هورمونهای مغز با یکدیگر کار می کنند تا بتوانند مسائل خاصی مانند تشخیص الگو یا طبقه بندی داده ها را از طریق فرایند یادگیری حل نمایند .

۱-۱.شباهت با مغز

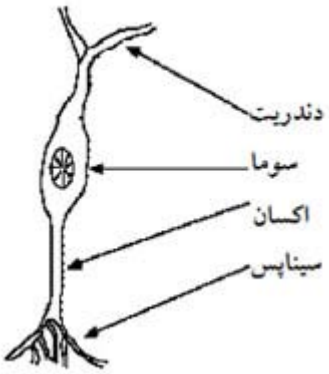
اگرچه مکانیسم‌های دقیق کارکرد مغز انسان (یا حتی جانوران) به‌طور کامل شناخته شده نیست، اما با این وجود جنبه‌های شناخته شده‌ای نیز وجود دارند که الهام بخش تئوری شبکه‌های عصبی بوده‌اند. به‌عنوان مثال، یکی ازسلول‌های عصبی، معروف به نرون (*Neuron*) است که دانش بشری آن را به‌عنوان سازنده اصلی مغز می‌انگارد. سلول‌های عصبی قادرند تا با اتصال به‌یکدیگر تشکیل شبکه‌های عظیم بدهند. گفته می‌شود که هر نرون می‌تواند به هزار تا ده هزار نرون دیگر اتصال یابد (حتی در این مورد عدد دویست هزار هم به عنوان یک حد بالایی ذکر شده است).

قدرت خارق‌العاده مغز انسان از تعداد بسیار زیاد نرون‌ها و ارتباطات بین آنها ناشی می‌شود.

ساختمان هر یک از نرون‌ها نیز به‌تنهایی بسیار پیچیده است. هر نرون از بخش‌ها و زیرسیستم‌های زیادی تشکیل شده است که از مکانیسم‌های کنترلی پیچیده‌ای استفاده می‌کنند. سلول‌های عصبی می‌توانند از طریق مکانیسم‌های الکتروشیمیایی اطلاعات را انتقال دهند. برحسب مکانیسم‌های به‌کاررفته در ساختار نرون‌ها، آنها را به بیش از یکصدگونه متفاوت طبقه‌بندی می‌کنند. در اصطلاح فنی، نرون‌ها و ارتباطات بین آنها، فرایند دودویی(*Binary*)، پایدار (*Stable*) یا همزمان (*Synchronous*) محسوب نمی‌شوند.

در واقع، شبکه‌های عصبی شبیه‌سازی شده یا کامپیوتری، فقط قادرند تا بخش کوچکی از خصوصیات و ویژگی‌های شبکه‌های عصبی بیولوژیک را شبیه‌سازی کنند. در حقیقت، از دید یک مهندس نرم‌افزار، هدف از ایجاد یک شبکه عصبی نرم‌افزاری، بیش از آنکه شبیه‌سازی مغز انسان باشد، ایجاد مکانیسم دیگری برای حل مسائل مهندسی با الهام از الگوی رفتاری شبکه‌های بیولوژیک است.

۱-۲.روش کار نرون‌ها



(*Neural Networks*)

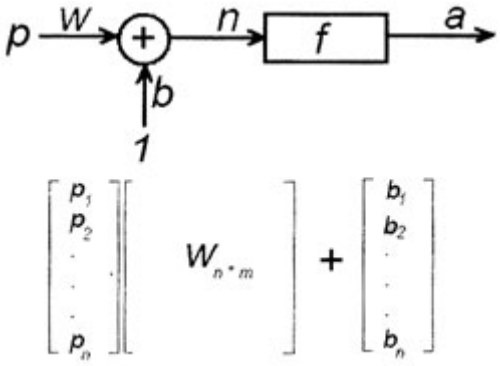
در شکل یک، نمای ساده شده‌ای از ساختار یک نرون بیولوژیک نمایش داده شده است. به‌طور خلاصه، یک نرون بیولوژیک، پس از دریافت سیگنال‌های ورودی (به شکل یک پالس الکتریکی) از سلول‌های دیگر، آن سیگنال‌ها را با یکدیگر ترکیب کرده و پس از انجام یک عمل (*operation*) دیگر بر روی سیگنال ترکیبی، آن را به‌صورت خروجی ظاهر می‌سازد.

همان‌طور که در تصویر مشاهده می‌کنید، نرون‌ها از چهار بخش اصلی ساخته شده‌اند. دندریت‌ها (*Dendrite*)، سوما (*Soma*)، اکسان (*Axon*) و بالاخره، سیناپس (*synapse*) دندریت‌ها، همان اجزایی هستند که به‌شکل رشته‌های طویل از مرکز سلول به اطراف پراکنده می‌شوند. دندریت‌ها نقش کانال‌های ارتباطی را برای انتقال‌دادن سیگنال‌های الکتریکی به مرکز سلول بر عهده دارند. در انتهای دندریت‌ها، ساختار بیولوژیکی ویژه‌ای به‌نام سیناپس واقع شده است که نقش دروازه‌های اتصالی کانال‌های ارتباطی را ایفا می‌کند. در واقع سیگنال‌های گوناگون از طریق سیناپس‌ها و دندریت‌ها به مرکز سلول منتقل می‌شوند و در آنجا با یکدیگر ترکیب می‌شوند. عمل ترکیب که به آن اشاره کردیم، می‌تواند یک عمل جمع جبری ساده باشد. اصولاً اگر چنین نیز نباشد، در مدل‌سازی ریاضی می‌توان آنرا یک عمل جمع معمولی در نظر گرفت که پس از آن تابع ویژه‌ای بر روی سیگنال اثر داده می‌شود و خروجی به شکل سیگنال الکتریکی متفاوتی از طریق اکسان (و سیناپس آن) به سلول‌های دیگر انتقال داده می‌شود.

البته تحقیقات جدید نمایانگر این واقعیت هستند که نرون‌های بیولوژیک بسیار پیچیده‌تر از مدل ساده‌ای هستند که در بالا تشریح شد. اما همین مدل ساده می‌تواند زیربنای مستحکمی برای دانش شبکه‌های عصبی مصنوعی (*Artificial Neural Network = ANN*) تلقی گردد و متخصصان گرایش شبکه‌های عصبی یا هوش مصنوعی می‌توانند با پیگیری کارهای دانشمندان علوم زیست‌شناسی، به بنیان‌گذاری ساختارهای مناسب‌تری در آینده دست بزنند.

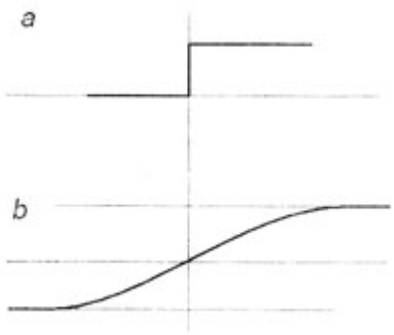
۱-۳.مدل ریاضی

در متون فنی برای نمایش مدل ساده‌ای که در بالا تشریح گردید، به‌طور معمول از شکلی مشابه شکل ۲ استفاده می‌شود. در این شکل کلاسیک، از علامت p برای نشان دادن یک سیگنال ورودی استفاده می‌شود. در واقع در این مدل، یک سیگنال ورودی پس از تقویت (یا تضعیف) شدن به اندازه پارامتر w، به‌صورت یک سیگنال الکتریکی با اندازه pw وارد نرون می‌شود. به‌جهات ساده‌سازی مدل ریاضی، فرض می‌شود که در هسته سلول عصبی، سیگنال ورودی با سیگنال دیگری به اندازه b جمع می‌گردد. در واقع سیگنال b خود به معنی آن است که سیگنالی به اندازه واحد در پارامتری مانند b ضرب (تقویت یا تضعیف) می‌شود. مجموع حاصل، یعنی سیگنالی به اندازه $pw + b$، قبل از خارج شدن از سلول تحت عمل یا فرآیند دیگری واقع می‌شود که در اصطلاح فنی به آن تابع انتقال (*Transfer Function*) می‌گویند. این موضوع در شکل به‌وسیله جعبه‌ای نمایش داده شده است که روی آن علامت f قرار داده شده است. ورودی این جعبه همان سیگنال $pw + b$ است و خروجی آن یا همان خروجی سلول، با علامت a نشانه گذاری شده است. در ریاضی، بخش آخر مدل‌سازی توسط رابطه $a = f(pw + b)$ نمایش داده می‌شود. پارامتر w یا همان ضریبی که سیگنال ورودی p در آن ضرب می‌شود، در اصطلاح ریاضی به نام پارامتر وزن یا *weight* نیز گفته می‌شود.



زمانی‌که از کنار هم قرار دادن تعداد بسیار زیادی از سلول‌های فوق یک شبکه عصبی بزرگ ساخته شود، شبکه‌ای در دست خواهیم داشت که رفتار آن علاوه بر تابع خروجی f، کاملاً به مقادیر w و b وابسته خواهد بود. در چنین شبکه بزرگی، تعداد بسیار زیادی از پارامترهای w و b باید توسط طراح شبکه مقداردهی شوند. این پروسه از کار، در اصطلاح دانش شبکه‌های عصبی، به فرآیند یادگیری معروف است. در واقع در یک آزمایش واقعی، پس از آن‌که سیگنال‌های ورودی چنین شبکه بزرگی اتصال داده شدند، طراح شبکه با اندازه‌گیری خروجی و با انتخاب پارامترهای w و b به‌گونه‌ای که خروجی مطلوب به‌دست آید، شبکه را <آموزش> می‌دهد. به این ترتیب پس از آنکه چنین شبکه به ازای مجموعه‌ای از ورودی‌ها برای ساختن خروجی‌های مطلوب <آموزش> دید، می‌توان از آن برای حل مسائلی که از ترکیب متفاوتی از ورودی‌ها ایجاد می‌شوند، بهره برد.

تابع f می‌تواند بر حسب کاربردهای گوناگون به‌طور ریاضی، به شکل های متفاوتی انتخاب شود. در برخی از کاربردها، پاسخ مسائل از نوع دودویی است. مثلاً مسأله به‌گونه‌ای است که خروجی شبکه عصبی باید چیزی مانند <سیاه> یا <سفید> (یا <آری> یا <نه> باشد. در واقع چنین مسائلی نیاز به آن دارند که ورودی‌های دنیای واقعی به مقادیر گسسته مانند مثال فوق تبدیل شوند. حتی می‌توان حالاتی را در نظر گرفت که خروجی دودویی نباشد، اما همچنان گسسته باشد. به عنوان مثال، شبکه‌ای را در نظر بگیرید که خروجی آن باید یکی از حروف الفبا، مثلاً از بین کاراکترهای اسکی (یا حتی یکی از پنجاه هزار کلمه متداول زبان انگلیسی) باشد. در چنین کاربردهایی، روش حل مسئله نمی‌تواند صرفاً بر جمع جبری سیگنال‌های ورودی تکیه نماید. در این کاربردها، ویژگی‌های خواسته شده فوق، در تابع خروجی یا تابع انتقال f گنجانیده می‌شوند. مثلاً اگر قرار باشد خروجی فقط یکی از مقادیر صفر یا یک را شامل شود، در این صورت می‌توان تابع خروجی شبکه عصبی را به شکل بخش a شکل شماره ۳ انتخاب کرد. در این حالت، خروجی چنین شبکه‌ای فقط می‌تواند بر حسب ورودی‌های متفاوت، مقدار یک یا صفر باشد.



در گروه دیگری از مسائلی که حل آن‌ها به شبکه‌های عصبی واگذار می‌شود، خروجی شبکه عصبی الزاماً بین مقادیر معلوم و شناخته شده واقع نمی‌شود. مسائلی از نوع شناسایی الگوهای تصویری، نمونه‌ای از چنین مواردی محسوب می‌شوند. شبکه‌های عصبی در این موارد، باید به‌گونه‌ای باشند که قابلیت تولید مجموعه نامتناهی از پاسخ‌ها را داشته باشند. رفتار حرکتی یک روبات نمونه‌ای از <هوشی> است که چنین شبکه‌های عصبی می‌توانند فراهم آورند. اما در چنین شبکه‌هایی هم لازم خواهد بود که خروجی بین مقادیر مشخصی محدود شده باشد (موضوع محدود شدن خروجی بین دو مقدار حدی ماکزیمم و مینیمم را در اینجا با موضوع قبلی اشتباه نگیرید. در این مورد خروجی مسأله اساساً گسسته نیست و حتی در بین چنین مقادیر حدی، می‌توان به تعداد نامتناهی خروجی دست یافت). اهمیت این موضوع زمانی آشکار می‌شود که از مثال واقعی کمک گرفته شود. فرض کنید قراراست از شبکه عصبی برای کنترل حرکت بازوی یک روبات استفاده شود. در صورتی‌که خروجی یک شبکه عصبی برای کنترل نیروی حرکتی به‌کار گرفته شود، طبیعی خواهد بود که اگر خروجی شبکه محدود نشده باشد، ممکن است بازوی روبات بر اثر حرکت بسیار سریع، به خود و یا محیط اطراف آسیب برساند. در چنین مواردی ممکن است از تابع انتقال به‌شکل بخش b شکل شماره ۳ استفاده شود.

قبل از آنکه به بخش دیگری از موضوع شبکه‌های عصبی بپردازیم، باید یک نکته را یادآوری کنیم که همان‌طور که در ابتدای این بخش تشریح شد، سلول‌های عصبی دارای ورودی‌های متعددی هستند و خروجی آنها نیز الزاماً محدود به یک خروجی نیست. بر این اساس زمانی که بخواهیم از مدل‌سازی ریاضی برای مدل کردن یک سلول عصبی استفاده کنیم، به‌جای آن‌که همانند شکل ۲ از یک ورودی p و یک خروجی a استفاده کنیم، از یک بردار p و یک بردار a سخن می‌گوییم. بدین ترتیب بدون آنکه نیاز به اعمال تغییری در این تصویر داشته باشیم، می‌توانیم از آن برای مدل‌سازی سلولی با n ورودی ($p1,p2,p3 \ldots pn$) و به همین ترتیب m خروجی ($a1,a2,am$) استفاده کنیم. توجه داشته باشید که در این صورت، تعداد عناصر b و w
نیز به تناسب افزایش می‌یابند و هر یک به n عدد افزایش می‌یابند.

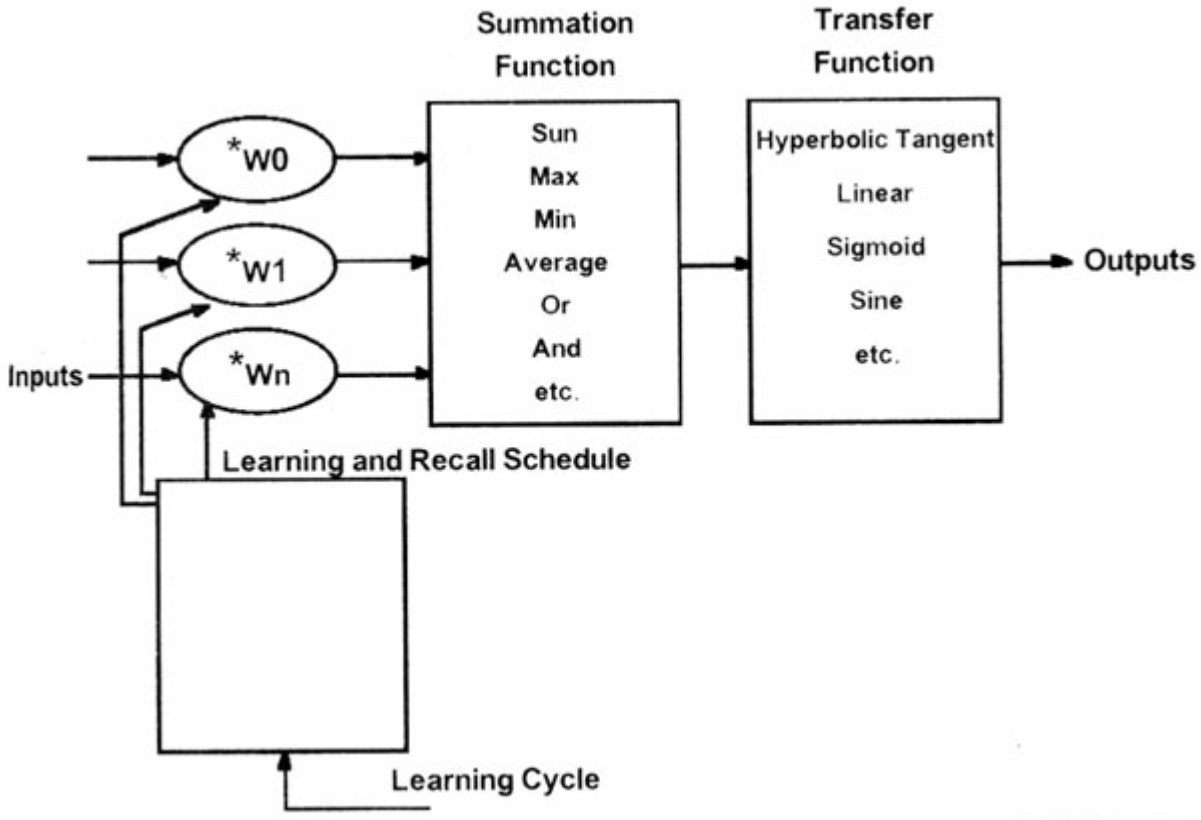
۲.عملیات شبکه‌های عصبی

تا اینجا تمام توجه ما معطوف ساختار درونی یک نرون مصنوعی یا المان پردازشی بود. اما بخش مهم دیگری در مراحل طراحی یک شبکه عصبی نیز وجود دارد. در واقع هنر یک طراح شبکه‌های عصبی می‌تواند در چگونگی ترکیب نرون‌ها در یک شبکه (*neuran Clustering*)، متجلی شود. علوم بیولوژی نشان داده‌اند که کلاسترینگ نرون‌های شبکه‌های عصبی را می‌توان با اغماض زیاد، مدل‌های الکترونیکی از ساختار عصبی مغز انسان نامید. مکانیسم فراگیری و آموزش مغز اساساً بر تجربه استوار است. مدل‌های الکترونیکی شبکه‌های عصبی طبیعی نیز بر اساس همین الگو بنا شده‌اند و روش برخورد چنین مدل‌هایی با مسائل، با روش‌های محاسباتی که به‌طور معمول توسط سیستم‌های کامپیوتری در پیش گرفته شده‌اند، تفاوت دارد. می‌دانیم که حتی ساده‌ترین مغزهای جانوری هم قادر به حل مسائلی هستند که اگر نگوییم که کامپیوترهای امروزی از حل آنها عاجز هستند، حداقل در حل آنها دچار مشکل می‌شوند. به عنوان مثال، مسائل مختلف شناسایی الگو، نمونه‌ای از مواردی هستند که روش‌های معمول محاسباتی برای حل آنها به نتیجه مطلوب نمی‌رسند. درحالی‌که مغز ساده‌ترین جانوران به‌راحتی از عهده چنین مسائلی بر می‌آید. تصور عموم کارشناسان *IT* بر آن است که مدل‌های جدید محاسباتی که بر اساس شبکه‌های عصبی بنا می‌شوند، جهش بعدی صنعت *IT* را شکل می‌دهند. تحقیقات در این زمینه نشان داده است که مغز، اطلاعات را همانند الگوها (*pattern*) ذخیره می‌کند. فرآیند ذخیره‌سازی اطلاعات به‌صورت الگو و تجزیه و تحلیل آن الگو، اساس روش نوین محاسباتی را تشکیل می‌دهند. این حوزه از دانش محاسباتی (*computation*) به هیچ وجه از روش‌های برنامه‌نویسی سنتی استفاده نمی‌کند و به‌جای آن از شبکه‌های بزرگی که به‌صورت موازی آرایش شده‌اند و تعلیم یافته‌اند، بهره می‌جوید. در ادامه این نوشته به این واژگان که در گرایش شبکه‌های عصبی، معانی ویژه‌ای دارند، بیشتر خواهیم پرداخت

۲-۱.پیاده‌سازی‌های الکترونیکی نرون‌های مصنوعی

نرم‌افزارهایی که در آن‌ها از شبکه‌های عصبی استفاده شده است، نرون‌های شبکه را المان پردازنده (*Processing element*) می‌نامند. به‌طور معمول در محصولات نرم‌افزاری، المان‌های پردازنده قابلیت بسیار بیشتری از نمونه ساده‌شده‌ای که در بخش‌های پیشین تشریح کردیم، دارند. در شکل شماره ۴، نمایی با جزئیات بیشتر از یک نرون مصنوعی را نشان می‌دهد.

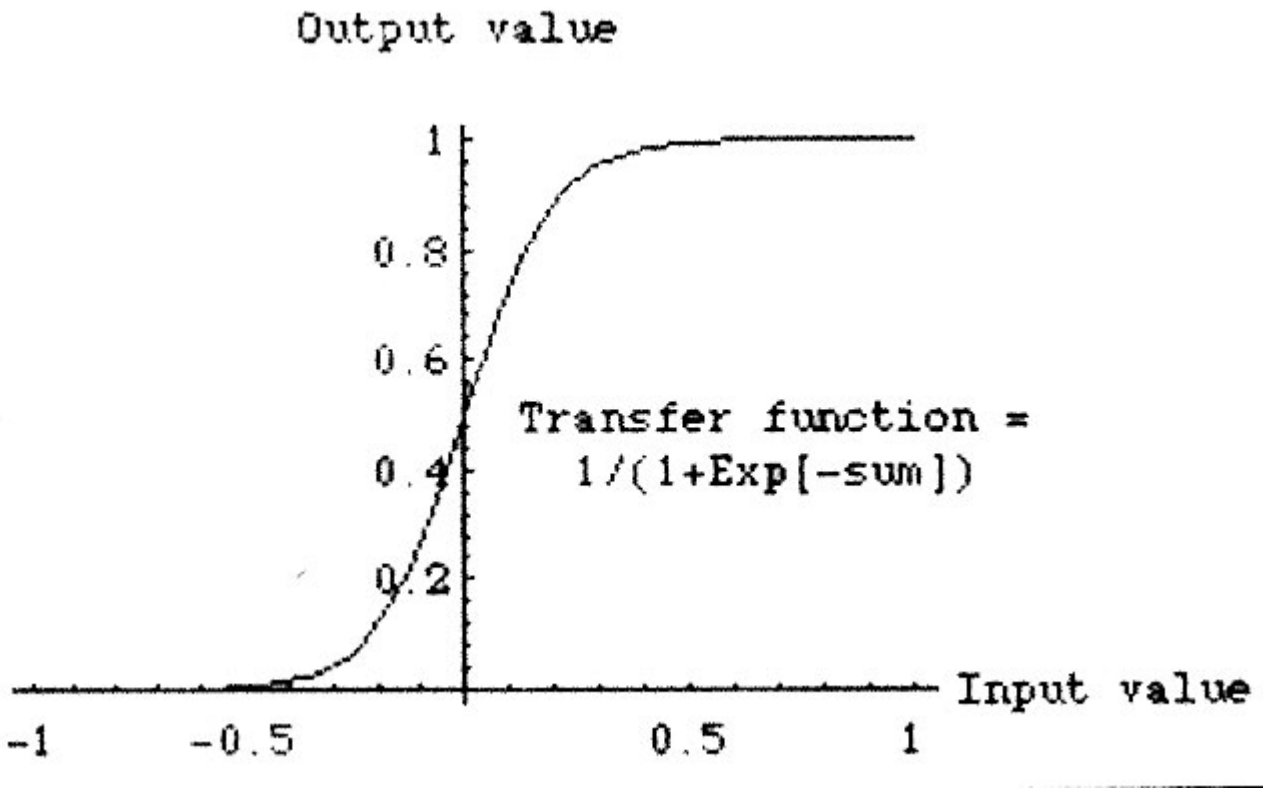
در این مدل، ورودی‌ها در نخستین گام، در ضریب وزنی (*weighting factor*) متناظر خود ضرب می‌شوند. در مرحله بعد، ورودی‌هایی که تغییر مقیاس داده‌اند وارد واحدی می‌شوند که در آن سیگنال‌های ورودی با هم ترکیب می‌شوند. به‌طور معمول عمل ترکیب در این واحد همان عمل جمع جبری است، اما در اصول، می‌توان در این واحد، ورودی‌ها را به روش‌های دیگری علاوه بر عمل جمع معمولی نیز با یکدیگر ترکیب کرد. مثلاً می‌توان به‌جای عمل جمع، از عمل متوسط گیری، انتخاب بزرگترین، انتخاب کوچک‌ترین، عمل *OR* یا *AND* منطقی هم استفاده کرد. در واقع هدف نهایی در این واحد آن است که از تعداد n ورودی، یک سیگنال خروجی برای ارائه به بخش‌های بعدی فرایند، به‌دست آید. انتخاب نوع <عمل> در این واحد، موضوع دقیقی است که کاملاً به کاربرد مسأله وابسته است



به‌طور معمول در نرم‌افزارهای تجاری، امکان انتخاب و حتی ساختن توابع گوناگون برای این واحد، از طرف نرم‌افزار به کاربران داده می‌شود. حتی می‌توان کاربردهایی یافت که در آنها، عمل ترکیب در این واحد، وابسته به زمان باشد و در زمان‌های گوناگون پردازش مسأله، عملیات مختلفی برای ترکیب کردن ورودی‌ها به‌کار برده شوند.

در هر صورت، پس از آنکه ورودی‌ها با یکدیگر ترکیب شدند، سیگنال حاصل به واحد دیگری که در آن تابع انتقال یا *Transfer Function* به سیگنال اعمال می‌شود، هدایت می‌گردد. خروجی این بخش، سیگنال‌های حقیقی خواهند بود. بدین ترتیب جعبه‌ای در دست خواهیم داشت که تعداد n عدد سیگنال ورودی را به m عدد سیگنال خروجی تبدیل می‌کند. در عمل توابع انتقالی که در بخش انتهایی نمودار شکل ۴ به‌کار برده می‌شوند، معمولاً یکی از توابع سینوسی، تانژانت هذلولی، *sigmoid* و نظایر این‌ها است. در تصویر شماره ۵، نمونه‌ای از یک تابع انتقال از نوع *sigmoid* نمایش داده شده است. همانطور که در این شکل مشاهده می‌کنید، این تابع انتقال، سیگنال خروجی واحد ترکیب را به سیگنال خروجی تبدیل می‌کند که مقدار (یا اندازه آن) بین صفر و یک می‌تواند باشد.

در عمل، سیگنالِ خروجی یک المان پردازنده می‌تواند برحسب نوع کاربرد، به المان‌های پردازشی دیگر و یا به اتصالات دیگر خارج از شبکه عصبی هدایت شود. در واقع تمامی شبکه‌های عصبی بر اساس ساختار المان‌های پردازشی فوق کار می‌کنند. در قسمت بعدی این مقاله به تشریح عملیات در شبکه‌های عصبی و آموزش این شبکه‌ها می‌پردازیم



شبکه عصبی مغز ما به‌گونه‌ای است که ما را قادر می‌سازد تا اطلاعات را به صورتی پویا، تعاملی و خودسامان (*Selforganizing*) پردازش کنیم. در شبکه‌های عصبی بیولوژیک، نرون‌ها در ساختاری سه بعدی به یکدیگر اتصال یافته‌اند. اتصالات بین نرون‌ها در شبکه‌های عصبی بیولوژیک آنقدر زیاد و پیچیده‌است که به هیچ وجه نمی‌توان شبکه مصنوعی مشابهی طراحی کرد. تکنولوژی مدارات مجتمع امروزی به ما امکان می‌دهد که شبکه‌های عصبی را در ساختارهای دو بعدی طراحی کنیم. علاوه بر این،

چنین شبکه‌های مصنوعی دارای تعداد محدودی لایه و اتصالات بین نرون‌ها خواهند بود. بدین ترتیب، این واقعیات و محدودیت‌های فیزیکی تکنولوژی فعلی، دامنه کاربردهای شبکه‌های عصبی مبتنی‌بر تکنولوژی سیلیکونی را مشخص می‌سازند.
ساختار شبکه‌های عصبی امروزی، از لایه‌های نرونی تشکیل شده است. در چنین ساختاری، نرون‌ها علاوه بر آنکه در لایه خود به شکل محدودی به یکدیگر اتصال داده شده‌اند، از طریق اتصال بین لایه‌ها نیز به نرون‌های طبقات مجاور ارتباط داده می‌شوند. در شکل ۱ نمونه‌ای از ساختار لایه‌ای یک شبکه عصبی مصنوعی نمایش داده شده است (تعداد اتصالات ممکن بین نرون‌ها را در چنین ساختاری با تعداد اتصالات بین نرون‌های مغز انسان، مقایسه کنید).

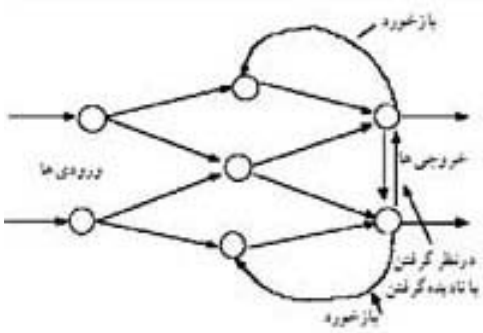
در این توپولوژی، گروهی از نرون‌ها از طریق ورودی‌های خود با جهان واقعی ارتباط دارند. گروه دیگری از نرون‌ها نیز از طریق خروجی‌های خود، جهان خارج را می‌سازند. در واقع این <جهان خارج> تصویری است که شبکه عصبی از ورودی خود می‌سازد یا می‌توان چنین گفت که جهان خارج <تصوری> است که شبکه عصبی از ورودی خود دارد. خلاصه آنکه در توپولوژی فوق، مابقی نرون‌ها از دید پنهان هستند.

تلاش محققان در زمینه شبکه‌های عصبی نشان داده است که شبکه‌های عصبی، چیزی بیشتر از یک مشت نرون که به یکدیگر اتصال داده شده‌اند، هستند. حتی گروهی از محققان سعی داشته‌اند که از اتصالات تصادفی برای ارتباط دادن نرون به یکدیگر استفاده کنند که در این زمینه به نتایج جالب توجهی دست نیافتند. امروزه مشخص شده است که در ساده‌ترین مغزهای بیولوژیک مانند مغز مارها هم ارتباطات بین نرون‌ها بسیار ساخت‌یافته است. در حال حاضر یکی از ساده‌ترین روش‌های ارتباط دهی نرون‌ها در شبکه‌های عصبی، آن است که ابتدا نرون‌ها در گروه‌های مشخصی به صورت لایه‌های نرونی سازمان‌دهی می‌شوند و پس از تامین ارتباطات بین‌نرونی در هر لایه، ارتباطات بین لایه‌ها نیز برقرار می‌شوند.

اگرچه در کاربردهای مشخصی می‌توان با موفقیت از شبکه‌های عصبی تک لایه استفاده کرد، اما رسم بر آن است که شبکه‌های عصبی حداقل دارای سه لایه باشند (همانطور که قبلاً اشاره شد، لایه ورودی، لایه خروجی و نهایتاً لایه پنهان یا لایه میانی).

در بسیاری از شبکه‌های عصبی، اتصالات بین‌نرونی به گونه‌ای است که نرون‌های لایه‌های میانی، ورودی خود را از تمام نرون‌های لایه پایینی خود (به طور معمول لایه نرون‌های ورودی) دریافت می‌کنند. بدین ترتیب در یک شبکه عصبی، سیگنال‌ها به تدریج از یک لایه نرونی به لایه‌های بالاتر حرکت می‌کنند و در نهایت به لایه آخر و خروجی شبکه می‌رسند. چنین مسیر در اصطلاح فنی *feed forward* نامیده

می‌شود. ارتباطات بین‌نرونی در شبکه‌های عصبی از اهمیت بسیار زیادی برخوردار هستند و به نوعی قدرت یک شبکه عصبی را تعیین می‌کنند. قاعده آن است که ارتباطات بین نرونی را به دو گروه تقسیم‌بندی می‌کنند. یک نوع از ارتباطات بین نرونی، به‌گونه‌ای هستند که باعث جمع شدن سیگنال در نرون بعدی می‌شوند. گونه دوم ارتباطات بین نرونی باعث تفریق سیگنال در نرون بعدی می‌شوند. در اصطلاح محاوره‌ای گروهی از ارتباطات انگیزش ایجاد می‌کنند و گروه دیگر ممانعت به عمل می‌آورند.



در مواردی، نرون مشخصی از شبکه عصبی تمایل دارد که سیگنال دیگر نرون‌های لایه خود را نادیده بگیرد. چنین حالتی به‌طور معمول در لایه خروجی ایجاد می‌شود. به عنوان مثال، در کاربردهای تشخیص متن (*OCR*)، فرض کنید که احتمال آنکه کاراکتر مورد شناسایی، حرف P باشد برابر با ۸۵ درصد تعیین شده است و به همین ترتیب احتمال آنکه کاراکتر مورد نظر حرف F باشد، ۶۵ درصد تخمین زده است. در این وضعیت، سیستم باید کاراکتری را برگزیند که دارای درصد احتمال بزرگ‌تر است. در نتیجه در این شبکه عصبی، نرون‌هایی که خروجی F را تجویز می‌کنند، باید نادیده گرفته شوند یا *inhibit* شوند. به چنین فرایندی، *lateral inhibition* گفته می‌شود.

نوع دیگری از ارتباط بین نرونی در شبکه‌های عصبی به ارتباط بازخورد یا *feedback* معروف است. در این نوع از ارتباطات، خروجی یک لایه نرونی به لایه قبلی (یا به لایه‌ای که چند مرحله پایینتر است) اتصال داده می‌شود. در شکل ۲ نمونه‌ای از یک شبکه عصبی نمایش داده شده که در آن از ارتباط بازخوردی استفاده شده است. در نرم‌افزارهای پیشرفته شبکه‌های عصبی، کاربر و طراح شبکه عصبی می‌تواند نوع ارتباطات بین نرون‌ها و لایه‌های آنها را تعیین کند.

۲-۲.آموزش شبکه‌های عصبی

تا اینجا از ساختار شبکه‌های عصبی صحبت کردیم. گفتیم که شبکه‌های عصبی می‌توانند بر اساس طراحی خود سیگنال‌های ورودی را پردازش کنند و به سیگنال‌های خروجی مورد نظر تبدیل نمایند. به طور معمول، پس از آنکه یک شبکه عصبی طراحی و پیاده‌سازی شد، باید پارامترهای w و b (که قبلاً معرفی کردیم) به ازای مجموعه‌هایی از سیگنال‌های ورودی، به‌گونه‌ای تنظیم شوند که سیگنال‌های خروجی شبکه خروجی مطلوب را تشکیل دهند. چنین فرایندی را آموزش دیدن شبکه عصبی می‌نامند

(در نخستین مرحله آموزش، مقادیر w و b به‌طور تصادفی انتخاب می‌شوند. زیرا تا این پارامترها مقدار نداشته باشند، شبکه عصبی قابل استفاده نخواهد بود) در حین آموزش دیدن شبکه عصبی (یعنی به تدریج همزمان با افزایش دفعاتی که مقادیر پارامترها برای رسیدن به خروجی مطلوب‌تر، تنظیم می‌شوند) مقدار پارامترها به مقدار حقیقی و نهایی خود نزدیک‌تر می‌شوند.

به‌طور کلی دو روش برای آموزش دادن شبکه‌های عصبی وجود دارد. روش *supervised* و روش *unsupervised*. روش نخست، شامل مراحلی است که در بخش قبل، به‌طور مختصر تشریح شد. اما در روش *unsupervised*، شبکه عصبی باید بدون کمک گرفتن از جهان خارج، بتواند کار آموزش را انجام دهد.

واقعیت آن است که در عمل از روش *supervised* و یا حداکثر از روش‌های ترکیبی استفاده می‌شود و فرایند آموزش *unsupervised* به شکل خالص تنها وعده‌ای است که شاید در آینده بتواند تحقق یابد.

در حال حاضر و در کاربردهای پیشرفته، از روش آموزش *unsupervised* برای ایجاد تنظیمات اولیه بر روی سیگنال‌های ورودی شبکه‌های عصبی استفاده می‌شود و باقی مراحل آموزش شبکه به روش *supervised* ادامه می‌یابد.

همان‌طور که قبلاً اشاره کردیم، در روش معمول آموزش شبکه‌های عصبی، از مجموعه شناخته‌شده‌ای از داده‌های ورودی و خروجی‌های متناظر آنها (*training set data*) برای آموزش دادن شبکه استفاده می‌شود. در چنین فرایندی، پس از اعمال مجموعه‌های داده‌های آموزشی، پارامترهای شبکه به تدریج به سمت مقادیر نهایی خود همگرا می‌شوند.

شبکه‌های عصبی قادر به یافتن الگوهایی در اطلاعات هستند که هیچکس، هیچگاه از وجود آنها اطلاع نداشته است.

بسته‌های نرم‌افزاری پیشرفته تولید و طراحی شبکه‌های عصبی، دارای ابزارهایی هستند که بر روند آموزش شبکه مدیریت می‌کنند. چنین ابزارهایی می‌توانند سرعت همگرایی پارامترهای شبکه را زیر نظر بگیرند و به عنوان مثال، اجازه دهند که پارامترهای یک شبکه مشخص، در طول چندین روز به دقت کافی و مورد نظر طراحان خود برسد.

در مواردی ممکن است که شبکه عصبی اصولاً موفق به فراگیری نشود. بدین معنی که پارامترهای شبکه پس از زمان‌های طولانی به مقدار مشخصی همگرا نشود. چنین مواردی ممکن است بر اثر ناکافی بودن داده‌های آموزشی و یا اصولاً نقص طراحی شبکه ایجاد شوند. حتی مواردی در عمل وجود دارند که شبکه عصبی مشخصی، بر اثر آموزش بیش از حد، اصطلاحا *over trained* شود. توجه داشته باشید که فرایند آموزش شبکه‌های عصبی فقط به ازای زیر مجموعه‌ای از داده‌هایی که قرار شبکه آنها را در کاربرد حقیقی خود پردازش کند، آموزش داده می‌شوند. درصورتی‌که تعداد داده‌های آموزشی یک شبکه عصبی بیش از اندازه زیاد باشد (در واقع از تمامی داده‌های مسئله برای آموزش دادن به شبکه استفاده شود)،

شبکه عصبی به جای آنکه آموزش ببیند، به حالتی می‌رسد که به آن حفظ کردن اطلاعات می‌گویند. در واقع به جای آنکه یک شبکه عصبی برای حل مسئله از هوش خود کمک بگیرد، از محفوظات خود استفاده می‌کند!

پس از آنکه یک شبکه عصبی به اندازه کافی آموزش دید، طراح یا کاربر شبکه می‌تواند پارامترهای شبکه را قفل کند (هر چند که در مواردی پارامترهای شبکه آزاد گذارده می‌شوند تا در طول کاربرد واقعی بازهم شبکه آموزش ببیند). در این مرحله شبکه عصبی برای کاربرد واقعی خود و حل مسائل آماده خواهد بود. در برخی از ابزارهای تولید و طراحی شبکه‌های عصبی، کل شبکه عصبی به همراه پارامترهای قفل شده آن، تبدیل به نرم‌افزار مستقلی (مثلاً یک فایل *dll*) می‌شوند که می‌توان از آن در پروژه‌های مشخصی استفاده کرد. در برخی از موارد دیگر، چنین شبکه‌هایی پس از آموزش دیدن، به شکل سخت‌افزاری در قالب یک مدار مجتمع (*IC*) به تولید انبوه یا نیمه انبوه می‌رسند

۲-۲-۱. آموزش *unsupervised* یا تطبیقی (*Adaptive*)

در مورد این روش آموزش گفتیم که شبکه عصبی بدون در اختیار داشتن داده‌های خروجی، در معرض آموزش قرار می‌گیرد. در واقع سیستم به تنهایی و بدون کمک خارجی باید با توجه به شکل سیگنال‌های خروجی خود، درباره درستی و نادرستی آنها تصمیم‌گیری نماید. در دنیای واقعی شرایط بسیار زیادی وجود دارند که در آنها مجموعه اطلاعات کافی برای آموزش دادن به سیستم فراهم نیستند. تحقیقات نظامی یکی از گرایش‌هایی است که به این موضوع توجه دقیقی دارد. به عنوان مثال گفته می‌شود که شرایط جنگی به دلیل فراوانی پارامترها و تکنیک‌های نظامی متغیر و پیشرفت‌های تکنولوژی نظامی، از نمونه مواردی است که در آنها به هیچ وجه نمی‌توان مجموعه داده‌های آموزشی کافی به دست آورد.

در این زمینه یکی از محققان شبکه‌های عصبی، به نام *Tuevo Kohonen* (از دانشگاه هلسینکی) فعالیتی جدی دارد. کوهنن با تحقیقات در ساختارهای عصبی غیرمتعارف، به پژوهش در این زمینه ادامه می‌دهد. کوهنن، نرون‌های شبکه‌عصبی را فیلدهای مختلفی تقسیم‌بندی می‌کند. در روش کوهنن، نرون‌های هر فیلد <مرتب توپولوژیک> یا *Topologically ordered* محسوب می‌شوند (توپولوژی نام شاخه‌ای از ریاضیات است که در آن نگاشت از یک فضا به فضای دیگر بدون تغییر مشخصه‌های هندسی، مورد بررسی قرار می‌گیرد). گروه‌بندی‌های سه‌بعدی که در ساختار مغز پستانداران یافت شده است، نمونه‌ای از مرتب‌سازی توپولوژیک محسوب می‌شوند. کوهنن معتقد است که فقدان ملاحظات توپولوژیک در مدل‌های عصبی امروزی، باعث می‌شود که شبکه‌های عصبی امروزی، مدل‌های ساده شده‌ای از شبکه‌های

عصبی واقعی موجود در مغز محسوب شوند. در هر صورت این حوزه از مبحث شبکه‌های عصبی، هنوز در مرحله تحقیقات آزمایشگاهی قرارداد و کاربرد واقعی نیافته است.

یادگیری در شبکه های عصبی به دو صورت می باشد :

۱- تحت نظارت *Supervised*))
۲- بدون دخالت انسانها (*Unsupervised*)

یادگیری در شبکه های عصبی رایج به شکل *Supervised* یا یادگیری تحت نظارت می باشد .
در واقع کار شبکه های عصبی مانند یادگیری بچه ها می باشد . با نشان دادن اشیاء ماهیت هر شیء برای کودک مشخص می شود .

به عنوان مثال کودک تصاویر انواع مختلف سگ را مشاهده می کند ، و در کنار آن اطلاعات ورودی تصاویر و صدا برای هر نمونه مطرح می گردد که این اطلاعات مربوط به یک نوع سگ می باشد و پس از مدتی او قادر خواهد بود یک نوع جدید از سگ را که قبلاً هرگز ندیده است ، شناسایی کند .

ANN شاخه ای از فیلد هوش مصنوعی و یا سیستمهای خبره می باشد که با منطق فازی مرتبط می باشد . یک شبکه عصبی مصنوعی می تواند به عنوان یک جعبه سیاه در نظر گرفته شود که قادر است الگوهای خروجی را پس از تشخیص الگوهای ورودی گزارش دهد .
شبکه های عصبی در واقع مثلثی هستند که سه ضلع مفهومی دارند :
۱- سیستم تجزیه و تحلیل داده ها
۲- نرون یا سلول عصبی
۳- شبکه یا قانون کار گروهی نرونها

شبکه های عصبی از دو بابت شبیه مغز عمل می کنند :
۱- مرحله ای موسوم به یادگیری دارند
نرون‌ها با پردازشگرهای شبکه به صورت غیر مستقیم به کانال‌های ارتباطی مرتبط هستند که وظیفه حمل داده ها را بر عهده دارند و تنها بر روی داده های محلی خود که به عنوان ورودی از طریق کانال های ارتباطی دریافت می دارند، عمل می کنند .این شبکه ها به صورت *Multilayer* می باشند که تعداد لایه های آن بستگی به پیچیدگی مسئله دارند و تعداد لایه ها و تعداد گره ها در هر لایه

مخفی از پارامترهایی است که توسط کاربر قابل تنظیم است .هر چه تعداد لایه ها بیشتر باشد سیستم قادر به درک پیچیدگی های بیشتری است .در این شبکه ها با پردازش موازی از طریق وزن ها سیناسپی داده ها راه خود را باز کرده و جلوی داده های *dump* (داده های دارای خطا یا بی ربط) گرفته می شود .طرز کار یک مدل سلول عصبی بدین صورت است که خطوط یا کانالهای ورودی ، سیگنالهای تحریکی یا مهاری را که همان پارامترهای تعریف کننده سیستم هستند به جسم سلولی یا گره های عصبی می آورند ، مثلاً غلظت یک ماده *mol/lit0.6* است. این پارامتر به عنوان یک سیگنال الکتریکی با شدت ./۶ به یک کانال ورودی می رود . در ابتدای هر کانال یک ضریب عددی (وزن سیناسپی) وجود دارد که شدت تحریک در آن ضرب می شود . اگر مثبت باشد یک سیگنال تحریکی و اگر منفی باشد یک سیگنال مهاری است؛ این سیگنال های تحریکی یا مهاری که از ورودی های مختلف به جسم سلولی می رسند ،با هم به صورت خطی جمع می گردد . اگر از میزان آستانه کمتر باشد سلول عصبی خاموش شده و در غیر این صورت *fire* (شلیک) می شود و جریان الکتریکی ثابتی در خروجی ایجاد می کند که به سلولهای دیگر وارد می شود . مشکل شبکه های عصبی این است که بتدریج به حفظ کردن الگوها می پردازند و قابلیت تجزیه و تحلیل آنها کم می گردد ، برای رفع این مشکل بایستی تعداد *node* های (گره ها) آنها کم گردد ، در این صورت شبکه مجبور به تعمیم می گردد .شبکه های عصبی مصنوعی برای حل مسائل پیچیده و یا مواردی که هیچ راه حل الگوریتمی وجود ندارد و یا بسیار پیچیده هستند مورد استفاده قرار می گیرد .چهار نوع هدف کلی توسط این شبکه های قابل پیگیری است ،که هر کدام بسته به نوع مجهولات در مواردی خاص قابل بهره گیری می باشد :

۳.اهداف شبکه عصبی

۳-۱. طبقه بندی

برای طبقه بندی ، داده های نمونه های مختلف را به شبکه می دهیم و نام گروه هر نمونه را به عنوان خروجی مشخص می کنیم ، پس از آموزش مناسب شبکه قادر خواهد بود با دریافت داده های مربوط به نمونه های جدید مشخص کند که این نمونه به کدام طبقه متعلق می باشد . به عنوان مثال میتوان پارامترهای آزمایشگاهی بیماران مبتلا به سرطان پروستات و افراد سالم را به عنوان ورودی و وضعیت فرد (سالم بودن یا سرطانی بودن) را به عنوان خروجی به شبکه داده در این صورت شبکه پس از یادگیری خواهد توانست پارامترهای فرد جدید را گرفته و سرطانی بودن او را پیشگویی کند .

۳-۲. تخمین تابع

زمانی که پارامترهای ورودی با تأثیرات پیچیده درسیستم پاسخی قابل اندازه گیری ایجاد می کنند ،شبکه می تواند آموزش بیابد تا این پاسخ را پیشگویی کند . به عنوان مثال شبکه می تواند پس از آموزش، با دریافت داده های مربوط به هر مولکول جدید در داروها ، شدت اثر آن را پیشگویی کند .

۳-۳. پیشگویی

اصطلاح پیشگویی در اینجا برای سری های زمانی بکاربرده می شود ؛ یعنی جایی که داده ها مربوط به نمونه های پیاپی هستند و داده های هر نمونه برای پیشگویی نمونه بعدی استفاده می شود . مانند پیشگویی وضعیت آتی بیمار بستری در بخش *CCU* .

۳-۴. خوشه کردن

این نوع کارکرد شبکه ها مربوط به یادگیری *Unsupervised* است . یعنی طبقه بندی داده ها بر حسب رفتار و برهم کنش های درونی آنها بدون داشتن الگو و یا فرضیه قبلی .

۳-۵. کاربرد شبکه های عصبی مصنوعی در علوم پزشکی

ANN در علوم پزشکی وداورویی نیز کاربرد بسیار گسترده ای دارد برخی کاربردهای آن عبارتند از :

۳-۵-۱. سیستم های تشخیص بیماری

ANN به صورت وسیعی در تشخیص بیماریها به کار گرفته شده است و این سیستمها قادرند برای تشخیص سرطان ، بیماریهای قلبی عروقی ، بیماری سل و عفونتهای سینوسی مورد استفاده قرار گیرند . از مزایای استفاده از *ANN* این است که فاکتورهایی چون خستگی، فرسودگی، وضعیت های عاطفی و یا تحت شرایط خاصی کارکردن روی آنها تاثیری ندارد.

۳-۵-۲. تجزیه و تحلیل های بیوشیمیایی

ANN به صورت وسیع و متنوعی در تجزیه وتحلیل نمونه های خون ، ادرار ، ردیابی سطح گلوکز در مبتلایان به دیابت ، تعیین سطح یون در مایعات بدن مورد استفاده قرار می گیرد.

۳-۵-۳.تجزیه و تحلیل تصویربرداری پزشکی

ANN در تجزیه وتحلیل تصاویر تومورها و *MRI* مورد استفاده قرار می گیرد.

۳-۵-۴. توسعه دارویی

ANN به عنوان ابزاری برای توسعه داروهای مرتبط با سرطان و ایدز مورد استفاده قرار می گیرد.
اگر چه در حال حاضر کاربرد شبکه های عصبی در دنیا مربوط به شبکه های تحت یادگیری است اما نوع دیگر شبکه ها که یادگیری *Unsupervised* دارند از هم اکنون مرزهای جدیدی را به سوی محققین گشوده اند و آرزوی یادگیری واقعی ماشینی ها بدون دخالت انسانها را برای محققین آرزویی دست یافتنی ساخته اند .

۴.تفاوت‌های شبکه‌های عصبی با روش‌های محاسباتی متداول و سیستم‌های خبره

گفتیم که شبکه‌های عصبی روش متفاوتی برای پردازش و آنالیز اطلاعات ارائه می‌دهند. اما نباید این گونه استنباط شود که شبکه‌های عصبی می‌توانند برای حل تمام مسائل محاسباتی مورد استفاده واقع شوند. روش‌های محاسباتی متداول همچنان برای حل گروه مشخصی از مسائل مانند امور حسابداری، انبارداری و محاسبات عددی مبتنی بر فرمول‌های مشخص، بهترین گزینه محسوب می‌شوند. جدول ۱، تفاوت‌های بنیادی دو روش محاسباتی را نشان می‌دهد

روش پردازش ترتیبی موازی توابع منطقی (*left brained*)از (*estault (right brained* روش فراگیری به کمک قواعد (*didactically*) با مثال (*Socratically*) کاربرد حسابداری، واژه پردازی، ریاضیات پردازش حسگرها، تشخیص گفتار، نوشتار سیستم‌های خبره، انشعابی از روش محاسباتی متداول محسوب می‌شود و در مواردی هم به آن نسل پنجم محاسبات نام داده‌اند (نسل اول از کلید و سیم‌بندی استفاده می‌کرد، نسل دوم با اختراع ترانزیستور ایجاد شد، نسل سوم از فناوری مدارات مجتمع استفاده می‌کرد، نسل چهارم با به وجود آمدن زبان‌های سطح بالا آغاز شد و نسل پنجم شامل هوش مصنوعی می‌شود). به طور معمول، یک سیستم خبره شامل دو بخش عمده می‌شود. یک بخش یا موتور استنتاجی و یک پایگاه دانایی (*Knowledge base*). موتور استنتاجی، بخشی است که رابط کاربر را مدیریت می‌کند و بر فایل‌ها و دسترسی به برنامه‌ها و برنامه‌ریزی کنترل دارد. پایگاه دانایی شامل اطلاعاتی در ارتباط با یک مسئله مشخص است. این پایگاه به متخصصان اجازه می‌دهد که قواعد فرایند مشخصی را تعریف نماید. چنین متخصصی نیازی به دانستن روش‌های برنامه‌نویسی نخواهد داشت. او تنها باید کاری که از کامپیوتر

می‌خواهد را درک کند و شناخت کافی از روش عمل سیستم داشته باشد. درواقع پوسته سیستم بخشی است که به کامپیوتر می‌گوید چه‌کار باید انجام دهد. برنامه لازم برای حل مسئله توسط خود سیستم تولید خواهد شد.
تلاش‌هایی که برای اجرایی کردن سیستم‌های خبره به کار گرفته شده‌اند، با مشکلات مشترکی مواجه بوده‌اند. با افزایش سطح پیچیدگی سیستم‌ها، منابع کامپیوتری مورد نیاز سیستم به شدت افزایش می‌یابند و سیستم با کندی بیش از حد روبرو می‌شود. در حقیقت تجربه نشان داده است که در وضعیت فعلی، سیستم‌های خبره تنها می‌توانند در مواقعی مفید واقع شوند که هدف محدود و مشخصی تعیین شده باشد.

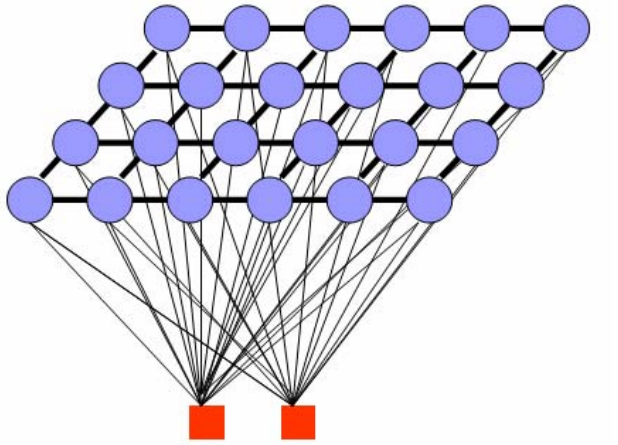
شبکه‌های عصبی در مسیری گام برمی‌دارند که ابزارها توانایی فراگیری و برنامه‌ریزی خود را داشته باشند. ساختارشبکه‌های عصبی به گونه‌ای است که قابلیت حل مسئله را بدون کمک فرد متخصص و برنامه‌ریزی خارجی داشته باشند. شبکه‌های عصبی قادر به یافتن الگوهایی در اطلاعات هستند که هیچ‌کس، هیچ‌گاه از وجود آنها اطلاع نداشته است.

درحالی‌که سیستم‌های خبره در عمل به موفقیت‌های بسیاری دست یافته‌اند، شبکه‌های عصبی در کاربردهایی همچون دید مصنوعی، تشخیص و تولید پیوسته گفتار، فراگیری ماشینی و نظایر آن با مشکلاتی روبرو بوده‌اند. در حال حاضر شبکه‌های عصبی کاملاً وابسته به سرعت پردازنده سیستم اجرا کننده هستند

۵.شبکه های خود سازمان ده کوهونن

در این بخش روش های فراگیری بدون سرپرست به ویژه نگاشتهای خود سازمان ده کوهونن را مورد بحث قرار می دهیم.در روش های پس انتشار خطا ملاحضه کردیم که آموزش با سرپرست به پاسخ های آموزشی خارجی (پاسخ های مطلوب) که برای کلیه داده های ورودی مجموعه آموزشی معلوم است تکیه دارد. لیکن در بسیاری از کاربردها بهتر است از شبکه بخواهیم که داده های آموزشی را به تنهایی طبقه بندی کند . برای این منظور دو فرض اصلی را برای شبکه باید در نظر بگیریم.اول اینکه عضویت در یک طبقه به طور کلی به معنای داشتن ویژگی های مشترک است. دیگر انکه شبکه می تواند این ویژگی های مشترک را در گسنره داده های ورودی تمیز دهد.باید توجه دااشت بردار ورودی ، بردار x عضو فضای R می باشد که هر یک از درایه های آن دارای یک چگالی احتمال است . در این فضای چگالی نمونه هایی را به تناوب و تصادف انتخاب کرده و به شبکه اعمال می کنیم بر اساس موقعیت بردار ورودی در فضای

R وزن های سلول ها طبق الگوریتمی که توضیح داده می شود ، تغییر می کند. این تغییر به نحوی انجام می گیرد که در نهایت، بردار های وزن مربوط به سلول ها ($w \epsilon R$) به طور یکنواخت در فضای چگالی احتمال ورودی توزیع می شود و بدین ترتیب شبکه با پراکندن سلول های خود در فضای ورودی چگالی احتمال را تخمین می زند . به کارگیری الگوریتم کوهونن اکثرا دو بعدی است. نکته اشکاری که باید به آن اشاره شود این است که در این شبکه نرون ها به عکس پرسپترون های چند لایه ای در لایه های متفاوت قرار نگرفته اند، بلکه در شبکه ای صاف و هموار ، مانند شکل زیر واقع شده اند.همه ورودی ها به تمام گره های خروجی متصل شده اند. باز خور تنها محدود به ارتباطات جانبی به نرون هایی است که در همسایگی یک نرون قرار دارند. هر گره در شبکه خود، در واقع یک گره محسوب میشود.



۵-۱.الگوریتم کوهونن

الگوریتم آموزشی گره های شبکه را به صورت همسایه های محلی منظم می کند به طوری که بتواند ویژگی های داده های ورودی را طبقه بندی کنند . نقشه توپوگرافی شبکه به طور خودکار با مقایسه دوره ای ورودی های هر گره با برداری که توسط آن گره در خطوط ارتباطی آن ذخیره شده است ، شکل می گیرد. هیچ جواب مطلوبی برای ورودی های آموزشی تعریف نمی شود. هرگاه ورودی های یک گره با بردار ذخیره شده آن گره مطابقت کند، آن ناحی شبکه به طور گزینشی بهینه می شود و به صورتی که نهایتا نمایانگر میانگینی از داده های آموزشی طبقه ای از داده ها شود. شبکه در حالی که ابتدا به طور تصادفی شکل گرفته است ، به تدریج خود را تنظیم می کند و به حالت پایداری می رسد که به طور موضعی ویژگی های داده های ورودی را نمایش می دهد .

الگوریتم شبکه کوهونن

۱ – مقادیر اولیه را تعیین کنید.

فرض کنید، $Wij(t)$ مقدار ضرایب وزنی از ورودی I به گره خروجی j در زمان t می باشد. میزان اولیه ضرایب وزنی را با مقایر کوچک تصادفی معیین کنید. شعاع همسایگی گره های خروجی j ، $Nj(0)$ ، را در ابتدا بزرگ تعیین کنند.

۲ – ورودی ها را به شبکه عرضه کنید.

۳ – فاصله ها را محاسبه کنید.

فاصله dj بین بردار ورودی و بردار خروجی هر گره j توسط فرمول زیر محاسبه می شود.

$$dj = \sum_{i=0}^{n-1} \left[Xi(t) - Wij(t) \right]^2$$

۴- کوتاه ترین فاصله را انتخاب می کنیم.

گره خروجی دارای کوتاه ترین فاصله را با علامت j^* مشخص می کنیم .

۵- ضرایب وزنی را اصلاح می کنیم.

ضرایب وزنی جدید با استفاده از فرمول زیر محاسبه می شود.

$$Wij(t+1)=Wij(t)+eta(t)*(Xi(t)-Wij(t))$$

عبارت $eta(t)$ همان ضریب بهره می باشد.($0 < eta(t) < 1$) که به تدریج با طول زمان کاهش می یابد. بنابراین ضریب بهره ضرایب وزنی به تدریج کند می شود. اندازه شعاع همسایگی $Nj^*(t)$ نیز به تدریج کاهش می یابد. بدین صورت محدوده بیشترین فعالیت به تدریج موضعی می گردد.

با رفتن به مرحله ۲ الگوریتم تکرار می شود.

۵-۱.نکاتی چند درباره الگوریتم کوهونن

۵-۱-۱.تشابه بردار ورودی با بردار وزن

بردار P وبردارهای دورنی آن هر دو در فضای R تعریف می شود ، لذا دارای یک جنسیت می باشد و با یکدیگر قابل مقایسه هستند. برای بررسی نزدیکی و دوری بردارهای w نسبت به بردار p در فضای R ، به یک معیار تشابه (یا یک معیار مسافت) نیاز داریم. این معیار می تواند فاصله اقلیدسی بین دو بردار باشد که به صورت زیر تعریف می شود.

$$dj = \sum_{i=0}^{n-1} \left[Xi(t) - Wij(t) \right]^2$$

۵-۱-۲.اصلاح ضرایب وزنی

در اصلاح ضرایب وزنی شبکه کوهونن نیاز به مشتق گیری نیست. با بررسی مجدد الگوریتم کوهونن مشاهده می کنیم که تغییرات وزنی متناسب با مقدار تفاوت بین بردار ورودی و بردار ضرایب وزنی است.

$$Wij(t+1)=Wij(t)+eta(t)*(Xi(t)-Wij(t))$$

ضرایب تناسب برابر با $eta(t)$ یعنی ضریب نرخ آموزش است در حالی که $0 < eta(t) < 1$ می باشد.این عبارت در طول زمان نرخ تطبیق را کاهش می دهد. اولین مرحله ، ایجاد نوعی نظم توپوگرافی در گره هایی است که ابتدا به طور تصادفی قرار گرفته اند. فرایند آموزش سعی می کند گره ها را در خوشه هایی قرار دهد که دامنه انواع طبقه ها را در داده های آموزشی نشان دهد . تغییرات در این مرحله در جهت قرار گرفتن گره ها روی نقشه توپوگرافی زیاد است. بنابراین ابتدا نرخ تطبیق بزرک تعیین می شود($0.5 < eta$) تا میزان تغییرات زیاد باشد و شبکه به سرعت به حالت تقریبا بهینه ای برسد. پس از اینکه نمادی تقریبی به دست آمد ، گره های مواضع محلی روی نقشه دقیق تر تنظیم شده و به بردار های آموزشی نزدیک می شود، برای این منظور باید اندازه تغییرات بسیار کوچکتر شود.بنابراین با پیشرفت زمان آموزش نرخ تطبیق کاهش می یابد. هرگاه ورودی آموزشی جدیدی به شبکه اعمال شود، ابتدا باید گره برنده را معین کرد.گره برنده گره ای خواهد بود که بردار ضرایب وزنی آن نزدیک ترین فاصله را با گره ورودی داشته باشد. میزان فاصله در اندازه گیری شباهت بردارها ، فاصله اقلیدسی است.نکات ضریفی در به کارگیری این میزان در روش کوهونن وجود دارد که باید به آنها دقت کنیم. نرم اقلیدسی یک بردار طول بردار را نشان می دهد . لیکن ما به طول بردارها چندان علاقه ای نداریم. آنچه مورد علاقه ما می باشد، جهت بردارها می باشد . به عبارت دیگر دو بردار را هنگامی شبیه می گوییم که هر دو بردار صرف نظراز طول آنها در یک جهت باشند. تنها راهی که می توان جهت دو بردار را با استفاده از فاصله اقلیدسی بررسی کنیم این است که ابتدا دو بردار را نرمالیزه کنیم. برای نرمالیزه کردن یک بردار عناصر یک بردار را بر طول بردار تقسیم می کنیم. بدین صورت برداررری به وجود می آید که طول آن برابر با واحد طول ، و جهت آن مشابه با جهت بردار اولیه است. در نتیجه مقایسه بردار ضرایب وزنی و بردار ورودی ، اکنون محدود به مقایسه جهت آن دو بردار خواهد شد. مزیت دیگر نرمالیزه کردن بردارها این است که زمان آموزش شبکه را با حذف یکی از عوامل متغییر فضای ضرایب وزنی کاهش می دهد . این عمل باعث می شود که نحوه قرار گرفتن بردارهای ضرایب وزنی در ابتدای آموزش نزدیک تر به حالت مطلوب باشد و در نتیجه زمان لازم برای اصلاح جهت آنها در طول آموزش کاهش یابد.

۵-۱-۳. تعیین مقادیر اولیه ضرایب وزنی

تا به حال پیشنهاد می کردیم که در ابتدای آموزش ضرایب وزنی را اعداد تصادفی کوچک و نرمالیزه شده قرار دهیم. لیکن موضوع به این سادگی نیست. چنانچه ضرایب وزنی در ابتدا واقعا تصادفی باشند، احتمال دارد که شبکه واقعا همگرا نشود و یا دوره ای آموزشی به کندی پیش رود. دلیل این حالت این است که بردارهای ورودی آموزشی به صورت خوشه های متمایز در مواضع محدودی از فضای الگوها بنابر

طبقه های خود قرار می گیرند. اگر بدار ضرایب وزنی گره های شبکه در ابتدا کاملا تصادفی باشند ، این احتمال می رود که تعداد زیادی از بردارهای ضرایب وزنی جهتی بسیار متفاوت با بردارهای آموزشی داشته باشند.این گره ها هرگز برنده هیچ کدام از بردارهای ورودی نخواهند شد و در تشکیل نقشه های توپولوژیکی شرکت نخواهند داشت. در نتیجه عملکرد طبقه بندی نقشه حاصل بسیار ضعیف خواهد بود ، زیرا نمی تواند ورودی ها را به خوبی تمیز دهد. به هر حال روش هایی برای تقریب این توزیع ها موجود است . یکی از روش ها ، انتخاب اولیه بردارهای ضرایب وزنی به صورتی است که تماما نرمالیزه بوده و بر هم منطبق باشند. به عبارت دیگر، تمام بردارها برابر تعیین می گردد . آنگاه تمام بردارهای آموزشی در ابتدای آموزش به صورت یک دسته بردار هم جهت با بردار اولیه گره ها به شبکه عرضه می شوند. این عمل به همه گره ها احتمال برابر می دهد که به بدار آموزشی نزدیک باشد.به تدریج که آموزش ادامه می یابد ، بردارهای ورودی به آرامی بهجهت اولیه خود برگردانده می شود. همچنین می توان به هر گره مقدار آستانه ای را اضافه نمود که تعدد موفقیت گره ها را کنترل کند، به طوری که اگر یک گره به طور مرتب برنده شد مقدار آستانه را افزایش داد و بدین طریق احتمال موفقیت بعدی را کاهش داد. در نتیجه سایر گره ها شانس موفقیت بیشتری را کسب کنند .

۵-۱-۴.انتخاب سلول برنده

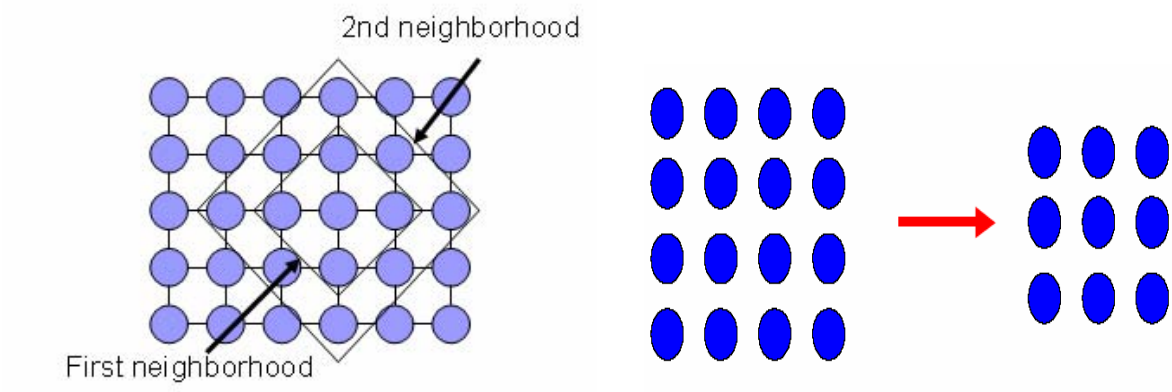
برای تعیین سلول برنده، همین که di کمینه را بیابیم کافی است. و سلولی که بردار وزنهای آن به بردار ورودی نزدیک تر باشد ، به عنوان سلول برنده شناخته می شود

۵-۱-۵. تعیین سلول هایی که به سمت ورودی حرکت می کنند

در این مورد نکته مهم آن است که از میان تمام سلولها تعدادی ، در حوالی سلول برنده ، به نحوی که سلول برنده در مرکز آنهاو بقیه در همسایگی آن قرار می گیرند، باید تغییر یابند انتخاب سلول های همسایه معمولا به دو روش صورت می پذیرد :اول انکه در یک شعاع همسایگی معین در اطراف سلول برنده ، همه سلول ها با یک ضریب برابر به سمت ورودی حرکت کنند . و دوم اینکه تمام سلول های موجود در شبکه ، ولی با ضرایب نا برابر ، به طرف ورودی حرکت کنند.این ضرایب نا برابر به نحوی هستند که در خود سلول برنده دارای بیشترین مقدار بوده و با دور شدن هر چه بیشتر از آن کمتر و کمتر می شود . در هر یک از این دو حالت زمان عمر شبکه تاثیر دارند در نوع اول شعاع همسایگی با زمان کاهش می یابد و در دومی ضرایب با گذشت زمان کمتر می شود. تفاوت عمده این دو روش در دو نکته زیر است.

اول اینکه روش نخست دارای یک تضاد شدید در لبه ناحیه همسایگی است ، ولی روش دوم چنین تغییر شدیدی را ندارد. این امر باعث بروز حرکت های شدید در روش اول شده و احتمال بروز خطا را افزایش می دهد. دوم اینکه در روش اول چون تعداد سلول هایی که تغییر حالت می یابد کمتر است ، سرعت

عملکـرد شـبکه بـالاتر خواهـد بـود، در حـالی کـه در روش دوم بایـد تمـامی سـلول هـا تغییـر حالـت بدهند.عملکرد در روش اول بدین صورت است که در ابتدا شعاع همسایگی تمامی گره هـا بـسیار بـزرگ تعیین می گردد. دامنه بزرگی می تواند به پهنای تمامی گره های شبکه باشد. وقتی گـره ای بـه عنـوان نزدیکترین گره به بردار ورودی برنده اعلام شود، ضرایب وزنی آن گره همراه با گره های موجود در شعاع همسایگی آن به مقدار یکسان اصلاح می شود. لیکن به تدریج که آموزش ادامه می یابد شعاع همسایگی به آرامی کاهش می یابد تا به حدی که قبلا تعیین شده است برسد.

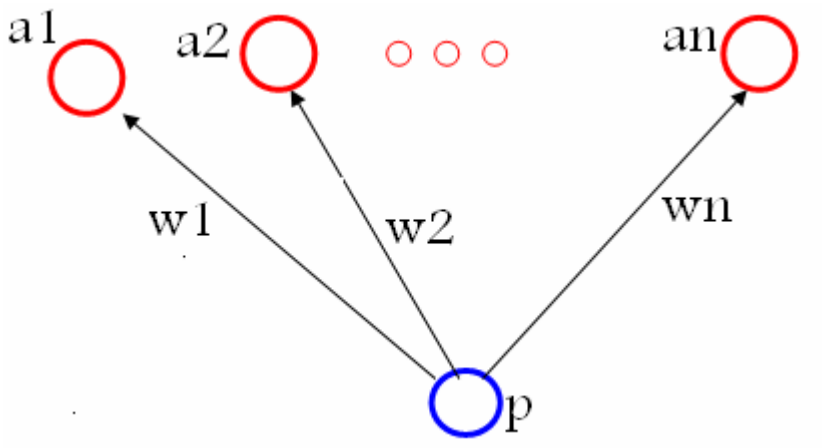


۵-۲.نحوه نمایش نگاشت

در اینجا نحوه نمایش وضعیت شبکه را برای سـه حالـت توپولـوژی یـک بعـدی بـا ورودی یـک بعـدی ، توپولوژی یک بعدی با ورودی دو بعدی و توپولوژی دو بعدی با ورودی دو بعدی مورد بررسـی قـرار مـی گیرد.

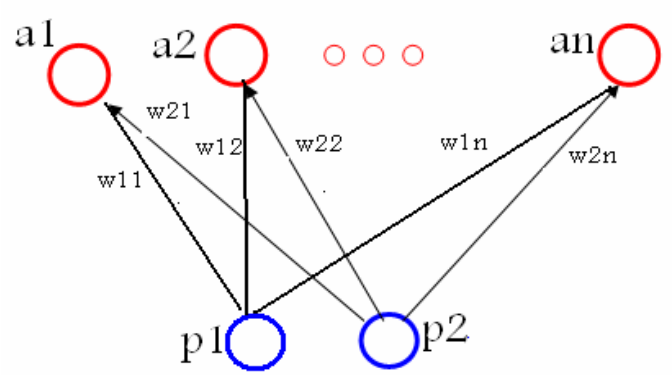
الف- توپولوژی یک بعدی با ورودی یک بعدی

در این شبکه یک بعدی هر سلول از دو طرف دارای همسایگی است و مـی تـوان سـلول هـا را از لحـاظ توپولوژی در یک خط در کنار یکدیگر نشان داد. همچنین هر سلول دارای یک وزن است کـه مـشخص کننده شدت اتصال بین آن سلول و تنها ورودی شبکه است.



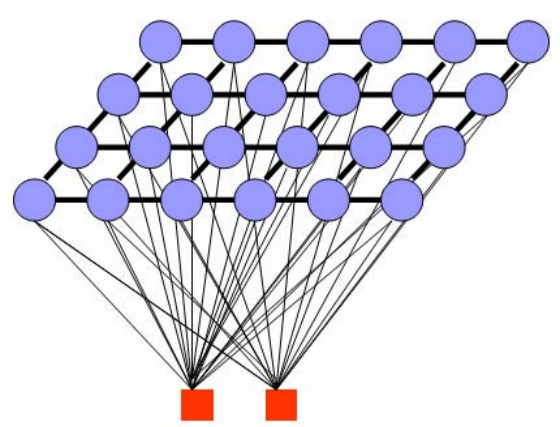
ب- توپولوژی یک بعدی با دو ورودی

در اینجا نیز چون توپولوژی یک بعدی استهر سلول در صفحه با یک خط به سلول مجاور خود وصل مـی شود.



ج- توپولوژی دو بعدی با دو ورودی

در اینجا طبیعتا هر سلول به چهار سلول همسایه خود وصل می شود. به این ترتیب بـه شـکل روبـرو در می آید



۵-۳.یک مسئله خواص در مورد SOM

مشکل نرون مرده و روش حل آن :

گاهی در شبکه وضعیتی پیش می آید که یک یا چند سلول عصبی دارای حالت هایی می شـوند کـه در طی بارها و بارها اعمال ورودی ، امکان برنده شدن را از دیگر سلول ها گرفته و خود برنده می شـوند. در این صورت بعضی از سلول ها از رقابت فاصله گرفته و تاثیر چندانی بر شبکه نخواهند داشت . این مشکل باعث می شود که در بعضی از قسمت های شکل که مورد نظر نیست، حالت بعضی از سلول ها جا مانده و تغییرات چندانی نداشته باشد. یک راه حل این مشکل آن است که تعداد دفعات برنده شـدن سـلول را در هنگام انتخاب سلول برتر دخالت دهیم، بدین ترتیب که سلول هـایی کـه بیـشتر برنـده مـی شـوند ، فرصت را در اختیار باقی سلول ها قرار دهند. برای نمونه اگر فرض کنیم تابع *f(ai)* یک تابع غیر نزولی از

i و i تعداد برنده شدن سلول از ابتدای حیات شبکه باشد، می توانیم برای انتخاب سلول برتر ، این تابع را با مسافت بین ورودی و حالت سلول جمع کنیم.

$$dj = dj + f(ai)$$

۶. چند کاربرد از شبکه کوهونن

۶-۱.ماشین تحریر صوتی

شاید بهترین راه نشان دادن ارزش یک ابتکار معرفی کاربردهای موفق آن باشد. کوهونن الگوریتم نگاشت ویژگی های خود را در مسئله شناخت گفتار که مداوم مورد توجه بوده است به کار گرفت. مسئله مورد بحث طبقه بندی فنوم های گفتاری در زمان واقعی است . فنوم ها مجموعه طبقه بندی کوچکی را تشکیل می دهند که نمونه های هر طبقه تفاوت های ظریفی با طبقات دیگر دارد این بدان معنی است که باید فقط تعداد محدودی گره ویژگی یاب در نقشه توپوگرافی شبکه تشکیل شود، که هر کدام گره های فراوان در همسایگی خود دارد که در فاصله محدود تنظیم شده است شناسایی گفتار از پیچیده ترین مسائل شناسایی الگوهاست . از نظر پردازش سیگنالها ، شکل موجی گفتار خوب تعریف نشده و پدیده ای پیچیده است . فنوم های گفتاری از نظر قدرت و شکل در افراد مختلف متفاوت است. حتی در یک فرد فنوم ها بسته به محتوای کلماتی که از آنها ساخته می شوند ، تغییر می کنند وفنوم های مختلف در بخش اعظم صورت موجی خود بر یکدیگر انطباق دارند. هدف اصلی کوهونن ساخت ماشین تحریر صوتی بود. یعنی ماشینی که بتواند گفتار دیکته شده را مستقیما تایپ کند . شبکه عصبی تنها بخشی از وظایف این سیستم را بر عهده داشت. در واقع از شبکه های عصبی منحصرا در مرحله طبقه بندی فنوم ها برای مقداری کردن برداری سیگنال های صوتی استفاده می شود. ماشین تحریر کوهونن از قسمت های زیر ساخته شده است :

۶-۱-۱.پیش پردازش

پیش پردازش از عناصر اصلی هر فناوری شبکه هاب عصبی است . هر نوع شبکه عصبی در صورتی که به آن داده های نمایانگری داده نشده یا آموزش کافی نبیند ، عملکرد ضعیفی خواهد داشت. هر چند شبکه های عصبی روش های نوینی را در تبدیل اطلاعات به کدهای توزیعی فراهم می کند، لیکن مانند هر روش الگو شناسی دیگر باید ابتدا از مرحله اساسی تعریف کردن درست و کافی مشخصه های داده ها گذر کند. کوهونن از روش های استاندارد پردازش سیگنال برای استخراج طیف فنوم ها در ورودی های صوتی استفاده می کند. امواج گفتاری ابتدا از یک میکروفن به یک فیلتر پایین گذر و دستگاه مبدل آنالوگ به دیجیتال *12* بیتی انتقال می یابد ، سپس با استفاده از یک تبدیل فوریه سریع *(FFT)* برای محاسبه داده های دیجیتال خروجی مبدل *ADC* محتوای طیفی فنوم ها استفاده می کند.

۶-۱-۲.شبکه های عصبی

شبکه های عصبی منحصرا در این مرحله طبقه بندی فنوم ها برای مقداری کردن برداری سیگنال ها به کار برده می شود.

۶-۱-۳.نقشه های کمکی

تلفظ برخی حروف ها مانند b ، x و g دارای طیفی بسیار ناپایدار است ، به طوری که در ابتدای طیف مقدار زیادی انرژی رها می شود وسپس با دوره ای از سکوت نسبی ادامه می یابد. کوهونن دریافت که شیوه معمول نقشه ی فنوتوپیک مناسب طبقه بندی این نوع از اصوات نمی باشد. راه حل او استفاده از نوعی نقشه کمکی ، به نام نقشه های گذرا می باشد. از نقشه های گذرا برای آموزش طیف این نوع فنوم ها استفاده می شود.

۶-۱-۴.پس پردازش

آخرین مرحله ماشین تحریر فنوتیک ، برگرداندن حالت فنوتیک به حالت نوشتاری می باشد. بیشتر خطا ها در این مرحله به سبب پدیده ای به نام هم نوایی صورت می گیرد. هم نوایی به تغییراتی اطلاق می گردد که در تلفظ فنوم ها به علت مظمون فنوم های هم جوار ایجاد می شود.برای مقابله با این پدیده کوهونن از سیستم قاعده پایه استفاده کرد که گرامر صحیح در ترجمه فنوتیک را بنا می کند. سیستم مذکور بسیار حجیم است .

۶-۲.متراکم سازی تصویر

مقصود از متراکم سازی اطلاعات تصویر کاهش هزینه ذخیره سازی و انتقال تصویر بدون آنکه کیفیت تصاویر حاصله تنزل نماید. برای این منظور بایستی تمام نقاط تصویر اولیه که با تعداد معینی از سطوح روشن بیان شده اند به سطوح دیگری تبدیل شوند که تعداد آنها کمتر از تعداد سطوح روشنایی تصویر اولیه است. برای این منظور می توان از شبکه کوهونن برای تخمین چگالی طیف روشنایی تصویر ورودی استفاده نمود . در این حالت تعداد نرون های شبکه برابر تعداد سطوح روشنایی دلخواه هستند که به کمک چگالی طیف روشنایی تصویر آموزش می یابند.

۶-۲-۱. برنامه متلب

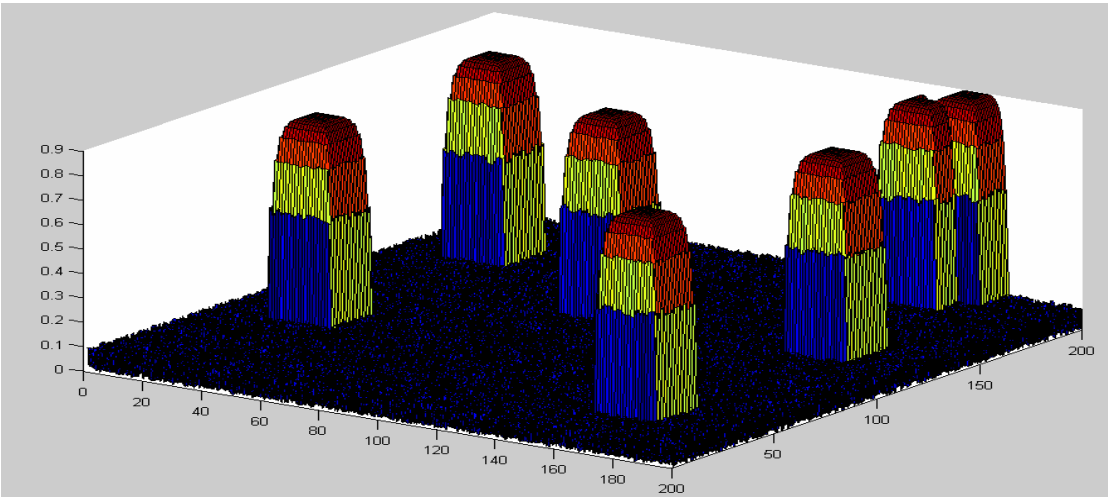
برنامه *train* مربوط به داده های ورودی به فرم عدد:

```
clear
n=200;  %output vector
m=200;  %output vector
i=7;    %input number
F=[ .1 .1 .1 .1 .1 .1 .9;    %input
   .1 .1 .1 .1 .1 .9 .1;
   .1 .1 .1 .1 .9 .1 .1;
   .1 .1 .1 .9 .1 .1 .1;
   .1 .1 .9 .1 .1 .1 .1;
   .1 .9 .1 .1 .1 .1 .1;
   .9 .1 .1 .1 .1 .1 .1]
out=[];
eta=.55;            %coefficient rate
w=.1*rand(m,n,i);    %weight
d=[];Step=1; N=10;   %step_eta=eta/(N-1);
for NN=1:7
   NN
   for sample=1:1
      x=F(:,NN);
      N=10;     %neighbor radial
      while N>2
         for I=1:m
            for J=1:n
               q=reshape(w(I,J,:),i,1);

               d(I,J)=(x-q)'*(x-q);
            end
         end
         [iu,ju]=find(d==min(d(:)));aa=[iu;ju];
         M=[iu+N iu-N ju+N ju-N];     % for boundary positions
         if M(1,4)<=0
            M(1,4)=1;
         end
         if M(1,3)>=n
            M(1,3)=n;
         end
         if M(1,2)<=0
            M(1,2)=1;
         end
         if M(1,1)>=m
            M(1,1)=m;
         end
         W_up=w(uint8(M(1,2):M(1,1)),uint8(M(1,4):M(1,3)),:);   %weight updating
         for I=1:size(W_up,1)
```

```
            for J=1:size(W_up,2)
                q=reshape(W_up(I,J,:),i,1);
                %q=W_up(I,J);
                q=q+eta.*(x-q);
                W_up(I,J,:)=q;
            end
        end
        w(M(1,2):M(1,1),M(1,4):M(1,3),:)=W_up;

        N=N-Step;
    end             % end of while
    out=[out aa];
  end
end
out
hold on   %show weight update for all output
ww=w(:,:,1);
surf (ww); figure(gcf)
ww=w(:,:,2);
surf (ww); figure(gcf)
ww=w(:,:,3);
surf (ww); figure(gcf)
ww=w(:,:,4);
surf (ww); figure(gcf)
ww=w(:,:,5);
surf (ww); figure(gcf)
ww=w(:,:,6);
surf (ww); figure(gcf)
ww=w(:,:,7);
surf (ww); figure(gcf)
hold off
save weight.mat w out
```



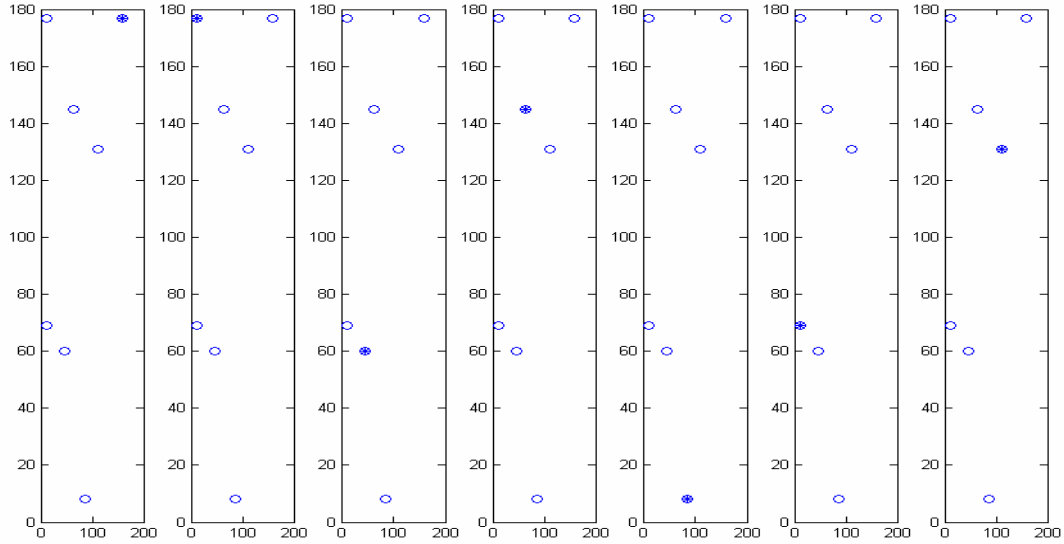
این شکل *update* شدن وزن ها را برای تمام ورودی ها نشان می دهد. همان طور که در شکل هم نشان داده شده این الگوریتم نرونی را به عنوان برنده انتخاب کـرده و شـعاع همـسایگی آن را *update* کـرده و

همچنان که شعاع همسایگی را کاهش می دهد آن را *update* نیز می کند. *Update* کردن شعاع همسایگی به این دلیل می باشد که چنانچه در قسمت *train* که به شبکه آموزش می دهیم که به ازای یک ورودی که مثلا سیب است وارد کلاس *a* شود، در قسمت *test* اگر چنانچه یک تکه از سیب را به این شبکه نشان دادیم ، به دلیل داشتن شعاع همسایگی *update* شده وارد کلاس دیگری نشود و وارد کلاس *a* شود.

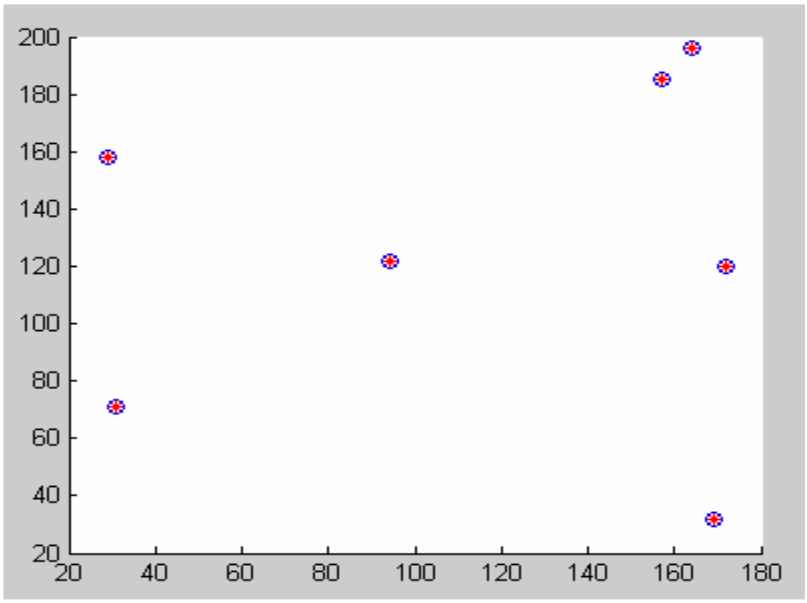
۶-۲-۲. برنامه *test* مربوط به داده های ورودی به فرم عدد:

```
clear
n=200;  %output vector
m=200;  %output vector
i=7;    %input number
F=[ .1 .1 .1  .1 .1  .1  .9;   %input
   .1 .1 .1  .1 .1  .9  .1;
   .1 .1 .1  .1 .9  .1  .1;
   .1 .1 .1  .9 .1  .1  .1;
   .1 .1 .9  .1 .1  .1  .1;
   .1 .9 .1  .1 .1  .1  .1;
   .9 .1 .1  .1 .1  .1  .1];
load weight.mat w out
F=F+0.8*rand(size(F));    %input whit noise
test=[];
for NN=1:7
   for sample= 1:1
      x=F(:,NN);
      for I=1:m
         for J=1:n
            for kk=1:7
               q(kk)=w(I,J,kk);
            end
            d(I,J)=norm(x'-q);
         end
      end
      [iu,ju]=find(d==min(d(:)));aa=[iu;ju];
      X=aa(1,1).*ones(length(out),1);
      Y=aa(2,1).*ones(length(out),1);
      X=[X';Y'];
      dis=(out-X).^2;
      d=sqrt(dis(1,:)+dis(2,:));
      [Val,Ind]=min(d);
      test=[test out(:,Ind)];
   end
end
out
test
O=[];
```

```
for i=1:7
    a=out(1,:)==test(1,i);
    b=out(2,:)==test(2,i);
    c=a&b;[a,b]=find(c==1);
    O=[O,F(:,b)];
end
O
for i=1:7    %showing win node for all inputs
hold on
subplot(1,7,i),plot(test(2,i),test(1,i),'*')
hold on
subplot(1,7,i),plot(out(2,:),out(1,:),'o')
end
```



شکل بالا نرونهای پیروز را برای تمام ورودی ها نشان می دهد.



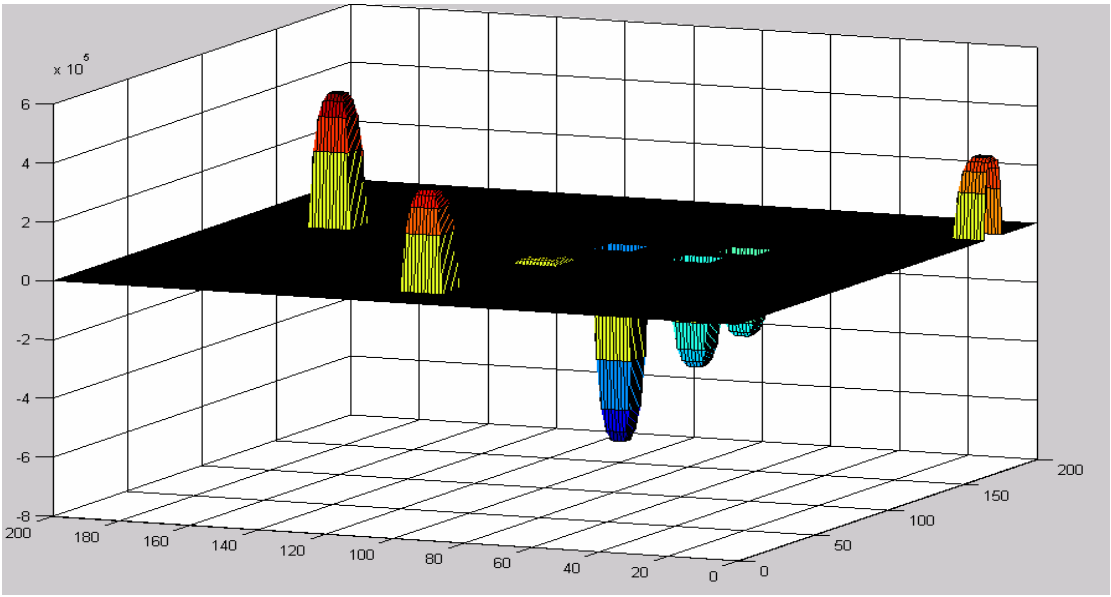
شکل بالا نرون های پیروز را برای ورودی های عدد نشان می دهد . دایره ها*(o)* نرون های پیـروز مربـوط به خروجی های *train* می باشدو ستاره ها*(*)* نرون های پیروز مربوط به تست می باشد.

۶-۲-۳. برنامه *train* مربوط به داده های ورودی به فرم تصویر

```
clear
n=200;  %output vector
m=200;  %output vector

load EDB33g.mat FNClass
Nclass=size(FNClass,2);
InpFea=FNClass(1).Fea;
i=size(InpFea,1);
out=[];
eta=.55;            %coefficient rate
w=.1*rand(m,n,i);    %weight
d=[];Step=1; N=5;   %step_eta=eta/(N-1);
alfa=zeros(n,m);
for NN=1:7
   NN
   for sample=2
      x=FNClass(NN).Fea(:,sample);
      %         x=length(OCR(1   ).Feature);
      N=5;     %neighbor radial
      while N>1
         for I=1:m
            for J=1:n
               q=reshape(w(I,J,:),i,1);
               d(I,J)=(x-q)'*(x-q);
            end
         end
         oo=d.*exp(alfa);
         [iu,ju]=find(oo==min   (oo(:)));aa=[iu(1);ju(1)];
         M=[iu(1)+N iu(1)-N ju(1)+N ju(1)-N];     % for boundary positions
         if ju(1)-N<=0
            M(1,4)=1;
         end
         if ju(1)+N>=n
            M(1,3)=n;
         end
         if iu(1)-N<=0
            M(1,2)=1;
         end
         if iu(1)+N>=m
            M(1,1)=m;
         end
         if N==5
            N1=M;
         end
         W_up=w(M(1,2):M(1,1),M(1,4):M(1,3),:);   %weight updating
         for I=1:size(W_up,1)
            for J=1:size(W_up,2)
```

```
            q=reshape(W_up(I,J,:),i,1);
            %q=W_up(I,J);
            q=q+eta.*(x-q);
            W_up(I,J,:)=q;
          end
        end
        w(M(1,2):M(1,1),M(1,4):M(1,3),:)=W_up;
        ww=w(:,:,1);
        surf (ww); figure(gcf)
        N=N-Step;    %eta=eta-step_eta;
      end            % end of while
      out=[out aa]
    end
    alfa(N1(1,2):N1(1,1),N1(1,4):N1(1,3),:)=alfa(N1(1,2):N1(1,1),N1(1,4):N1(1,3),:)+10;
end
out
save weight.mat w out
```

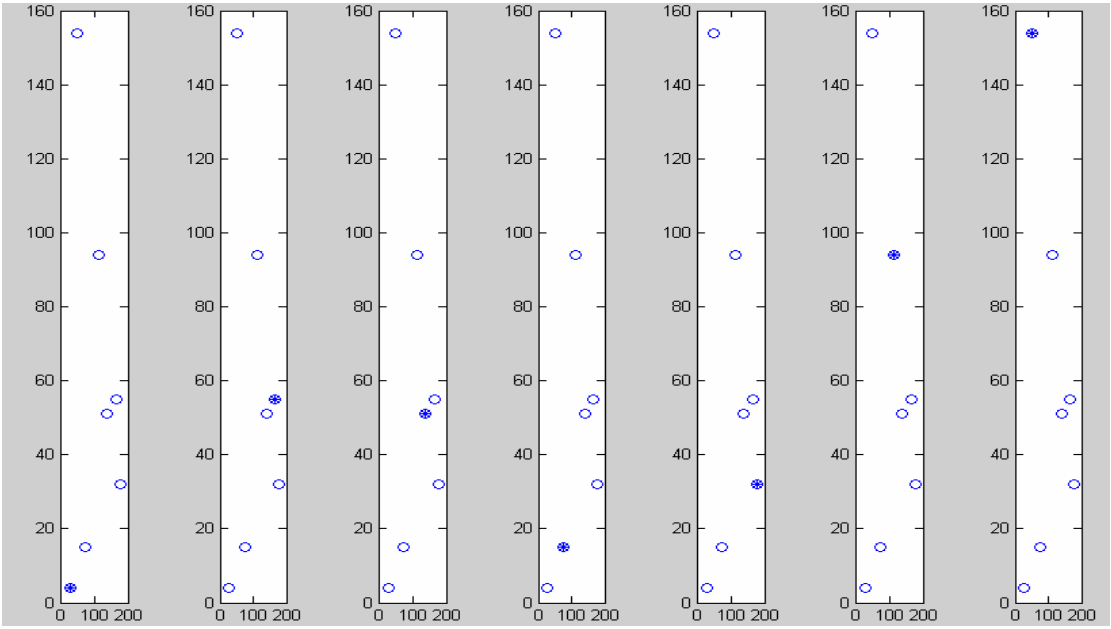


۶-۲-۴. برنامه *test* مربوط به داده های ورودی به فرم تصویر

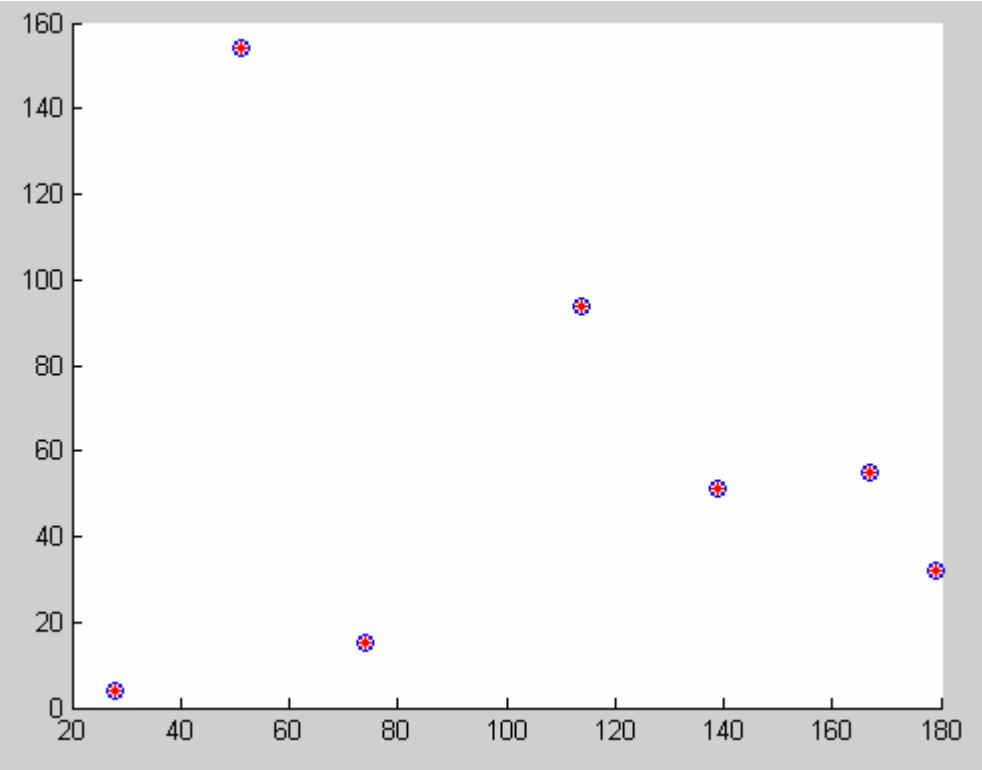
```
clear
n=200;  %output vector
m=200;  %output vector

load EDB33g.mat FNClass
Nclass=size(FNClass,2);
InpFea=FNClass(1).Fea;
i=size(InpFea,1);
load weight.mat w out
```

```
test=[];
for NN=1:7
   for sample=1
      x=FNClass(NN).Fea(:,sample);
      for I=1:m
         for J=1:n
            for kk=1:20
               q(kk)=w(I,J,kk);
            end
            d(I,J)=norm(x'-q);
         end
      end
      [iu,ju]=find(d==min(d(:)));aa=[iu;ju];
      X=aa(1,1).*ones(length(out),1);
      Y=aa(2,1).*ones(length(out),1);
      X=[X';Y'];
      dis=(out-X).^2;
      d=sqrt(dis(1,:)+dis(2,:));
      [Val,Ind]=min(d);
      test=[test out(:,Ind)];
   end
end
out
test
O=[];
for i=1:7
   a=out(1,:)==test(1,i);
   b=out(2,:)==test(2,i);
   c=a&b;[a,b]=find(c==1);
end
O
```



شکل بالا نرونهای پیروز را برای تمام ورودی ها نشان می دهد.



شکل بالا نرون های پیروز را برای ورودی های تصویر نشان می دهد . دایره ها*(o)* نرون های پیروز مربوط به خروجی های *train* می باشد و ستاره ها*(*)* نرون های پیروز مربوط به تست می باشد.

۷.منابع

مبانی شبکه های عصبی هوش محاسباتی تالیف دکتر محمد باقر منهاج .
شبکه های عصبی تالیف آر.بیل و تی. جکسون .

References:
[1] www.eca.ir
[2] www.nafttimes.com
[3] www.itj.sums.ac.ir
[4] www.cs.kent.ac.uk
[5] D.bianchi, R.Calogero , B.Tirozzi . kohonen neural networks and genetic classification , Roma.italy,2006