دانشگاه صنعتی شریف

دانشکده مهندسی برق

آشنایی مقدّماتی با شبکه‌های عصبی مصنوعی

گردآورنده:

امید صیّادی

دانشجوی دکترای مهندسی برق (بیوالکتریک)،
آزمایشگاه پردازش سیگنال‌های حیاتی و تصاویر پزشکی،
دانشکده مهندسی برق، دانشگاه صنعتی شریف
(osayadi@ee.sharif.edu)

اسفند ۱۳۸۷

مقدّمه:

شبکه‌های عصبی نوعی مدل‌سازی ساده‌انگارانه از سیستم‌های عصبی واقعی هستند که کاربرد فراوانی در حل مسائل مختلف در علوم دارند. حوزه کاربرد این شبکه‌ها آن‌چنان گسترده است که از کاربردهای طبقه‌بندی گرفته تا کاربردهایی نظیر درون‌یابی، تخمین، آشکارسازی و ... را شامل می‌شود. شاید مهمترین مزیّت این شبکه‌ها، توانایی وافر آن‌ها در کنار سهولت استفاده از آن‌ها باشد.

به موازات گسترش کاربردهای شبکه عصبی، نیاز به فراگیری آن و آشنایی با توانایی‌ها و قابلیت‌های آن رخ می‌نماید. در این راستا، تألیفات زیادی وجود دارد که می توان به آن‌ها استناد نمود امّا در اکثر قریب به اتفاق این خودآموزها، مبانی این شبکه‌ها با تفاصیل و جزئیات بسیار بیان شده است و مسلماً به عنوان قدم اول برای آشنایی، حجیم و وقت‌گیر به نظر می‌رسد. لذا بر آن شدیم تا با درنظر گرفتن نیاز دانشجویان، به ویژه در کاربردهای پروژه‌ای در مقطع کارشناسی، مقدماتی را جهت آشنایی با این شبکه‌ها تدوین کنیم به نحوی که به عنوان یک خودآموز، در پیمودن قدم‌های اولیه برای آشنایی و سپس تسلط بر مفاهیم مقدماتی آن‌ها، راهگشا باشد. بنابراین در این جزوه سعی کرده‌ایم مبانی مدل‌سازی و تحلیل یک سیستم با استفاده از شبکه‌های عصبی بیان شده و شبکه چندلایه پرسپترون، به عنوان یکی از پرکاربردترین شبکه‌ها مرور گردد. همچنین رویکرد در پیش‌رو بر این است که در کنار بیان اجمالی روابط ریاضی مورد نیاز، به نکات پیاده‌سازی و تکنیک‌های عملی استفاده از آن‌ها اشاره خواهیم کرد.

با توجه به گستردگی مطالب، در این نوشتار سعی بر آن شده است که از پرداختن به جزئیات غیرضروری پرهیز شود. لذا در این نسخه، تنها به مفاهیم مقدماتی و ضروری شبکه های عصبی مصنوعی خواهیم پرداخت، به این امید که دانشجویان عزیز بتوانند با فراگیری این نکات، مسیر فراگیری و به کار گیری این شبکه‌ها را ساده‌تر بپیمایند. در این راستا، رئوس مطالبی که به آن‌ها خواهیم پرداخت عبارتند از:

- مفهوم شبکه
- مدل ریاضی شبکه عصبی مصنوعی
- پرسپترون چند لایه
- آموزش شبکه به روش پس‌انتشار خطا
- روند شبیه‌سازی مسائل

مفهوم شبکه

یکی از روش‌های کارآمد در حل مسائل پیچیده، شکستن آن به زیرمسأله‌های ساده‌تر است که هر کدام از این زیربخش‌ها به نحو ساده‌تری قابل درک و توصیف باشند. در حقیقت یک شبکه، مجموعه‌ای از این ساختارهای ساده است که در کنار یکدیگر سیستم پیچیده نهایی را توصیف می‌کنند. شبکه‌ها انواع مختلفی دارند اما همگی آن‌ها از دو مؤلفه تشکیل می‌شوند:

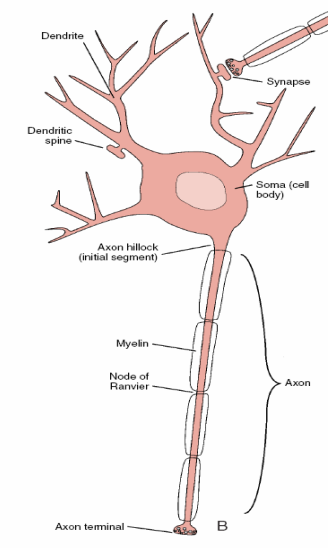
۱. مجموعه‌ای از گره‌ها؛ هر گره در حقیقت واحد محاسباتی شبکه است که ورودی‌ها را گرفته و برروی آن پردازش انجام می‌دهد تا خروجی بدست آید. پردازش انجام شده توسط گره می‌تواند از ساده‌ترین نوع پردازش‌ها نظیر جمع کردن ورودی‌ها تا پیچیده‌ترین محاسبات را شامل شود. در حالت خاص، یک گره می‌تواند خود، شامل یک شبکه دیگر باشد.

۲. اتصالات بین گره‌ها؛ این اتصالات نحوه گذر اطلاعات بین گره‌ها را مشخص می‌کند. در حالت کلی اتصالات می‌توانند تک‌سویه (Unidirectional) یا دوسویه (Bidirectional) باشند.

تعامل بین گره‌ها از طریق این اتصالات سبب بروز یک رفتار کلی از سوی شبکه می‌گردد که چنین رفتاری به تنهایی در هیچ یک از المان‌های شبکه دیده نمی‌شود. جامع بودن این رفتار کلی بر عملکرد موجود در هر گره سبب تبدیل شبکه به یک ابزار توانمند می‌شود. به عبارت دیگر، مجموعه ساده‌ای از المان‌ها وقتی در قالب یک شبکه باشند می‌توانند رفتاری از خود بروز دهند که هیچ یک از آن المان‌ها به تنهایی قادر به بروز چنین مشخصه‌ای نبود.

شبکه عصبی مصنوعی:

آنچنانکه بیان شد انواع مختلفی از شبکه‌ها وجود دارد. در این بین شبکه‌ای وجود دارد که گره را به عنوان یک نرون مصنوعی درنظر می‌گیرد. در اصطلاح، این چنین شبکه‌هایی را شبکه عصبی مصنوعی (Artificial Neural Network) یا به اختصار ANN می‌نامند.



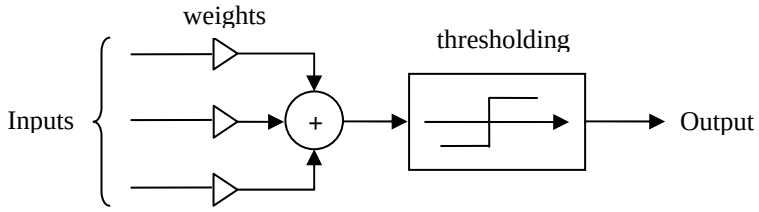
شکل ۱- یک نمونه عصب واقعی.

یک نرون مصنوعی در حقیقت مدلی محاسباتی است که از نرون‌های عصبی واقعی انسان، الهام گرفته است. نرون‌های طبیعی، ورودی خود را از طریق سیناپس دریافت می‌کنند. این سیناپس‌ها برروی دندریت‌ها یا غشاء عصب قرار دارند. در یک عصب واقعی، دندریت‌ها دامنه پالس‌های دریافتی را تغییر می‌دهند که نوع این تغییر در طول زمان یکسان نمی‌ماند و در اصطلاح، توسط عصب یاد گرفته می‌شود. اگر سیگنال دریافتی به حد کافی قوی باشد (از یک مقدار آستانه بیشتر شود)، عصب فعال شده و سیگنالی را در طول اکسون منتشر می‌کند. این سیگنال نیز به نوبه خود می‌تواند به یک سیناپس دیگر وارد شده و سایر اعصاب را تحریک کند. شکل ۱ یک نمونه عصب واقعی را نشان می‌دهد.

مدل ریاضی شبکه عصبی مصنوعی

به هنگام مدل کردن اعصاب، از پیچیدگی‌های آن‌ها صرف نظر می‌شود و تنها به مفاهیم پایه‌ای بها داده می‌شود، چرا که در غیر این صورت رویکرد مدل‌سازی بسیار دشوار خواهد شد. در یک نگاه ساده، مدل یک عصب باید شامل ورودی‌هایی باشد که در نقش سیناپس انجام وظیفه کنند. این ورودی‌ها در وزن‌هایی ضرب می‌شوند تا قدرت سیگنال را تعیین کنند. نهایتاً یک عملگر ریاضی تصمیم‌گیری می‌کند که آیا نرون فعال شود یا خیر و اگر جواب مثبت باشد، میزان خروجی را مشخص می‌سازد. بنابراین شبکه عصبی مصنوعی با استفاده از مدل ساده شده عصب واقعی به پردازش اطلاعات می‌پردازد. با توجه به این توضیحات، می‌توان مدل ساده‌ای برای توصیف یک نرون (یک گره در شبکه عصبی مصنوعی) پیشنهاد کرد. این مدل در شکل ۲ نشان داده شده است. جدای از ساده‌سازی‌های اعمال شده، تفاوت اصلی این مدل با واقعیت در این است که در شبکه واقعی، ورودی‌ها سیگنال‌های زمانی هستند حال آن‌که در این مدل، اعداد حقیقی ورودی‌اند.

در مدل ارائه شده در شکل ۲، تنوع‌های بسیاری وجود دارد. از جمله این‌که وزن‌های یک شبکه عصبی، که مقدار خروجی را منتقل می‌کنند، می‌توانند مثبت یا منفی باشند. از طرفی، توابع مورد استفاده برای آستانه‌گذاری می‌توانند بسیار متنوع باشند. از جمله مشهورترین این توابع می‌توان به تابع‌هایی نظیر $arcsin, arctan, sigmoid$ اشاره کرد. این توابع باید پیوسته و هموار بوده و مشتق‌پذیر باشند. همچنین تعداد گره های ورودی می‌تواند متغیر باشد. البته با زیاد شدن تعداد این گره‌ها، به وضوح تعیین وزن‌ها را با مشکل روبرو می‌کند. لذا باید به دنبال روش‌هایی برای حل این موضوع باشیم. روند تعین وزن‌های بهینه و تنظیم مقادیر آن‌ها عمدتاً به صورت بازگشتی انجام می‌شود. بدین منظور شبکه را با استفاده از قواعد و داده‌ها آموزش داده و با استفاده از قابلیت یادگیری شبکه، الگوریتم‌های متنوعی پیشنهاد می‌گردد که همگی سعی در نزدیک کردن خروجی تولید شده توسط شبکه به خروجی ایده‌آل و مورد انتظار دارند.



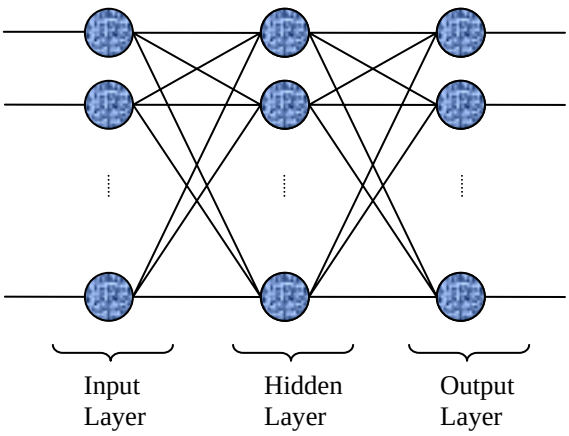
شکل ۲- مدل ریاضی ساده‌شده عصب واقعی.

پرسپترون چند لایه

هرچند نحوه مدل کردن نرون جزء اساسی‌ترین نکات کلیدی در کارآیی شبکه عصبی می‌باشد اما نحوه برقراری اتصالات و چیدمان (توپولوژی) شبکه نیز فاکتور بسیار مهم و اثرگذاری است. باید توجه داشت که توپولوژی مغز انسان آنقدر پیچیده است که نمی‌توان از آن به عنوان مدلی برای اعمال به شبکه عصبی استفاده نمود، چرا که مدلی که ما استفاده می‌کنیم، یک مدل ساده شده است در حالی که چیدمان مغز از المان‌های بسیار زیادی استفاده می‌کند.

یکی از ساده‌ترین و در عین حال کارآمدترین چیدمان‌های پیشنهادی برای استفاده در مدل‌سازی عصب‌های واقعی، مدل پرسپترون چندلایه (Multi layer perceptron) یا به اختصار MLP می‌باشد که از یک لایه ورودی، یک یا چند لایه پنهان و یک لایه خروجی تشکیل یافته است. در این ساختار، تمام نرون‌های یک لایه به تمام نرون‌های لایه بعد متصلند. این چیدمان اصطلاحاً یک شبکه با اتصالات کامل را تشکیل می‌دهد.

شکل ۳ شمای یک شبکه پرسپترون سه لایه را نشان می‌دهد. به سادگی می‌توان استنباط نمود که تعداد نرون‌های هر لایه، مستقل از تعداد نرون‌های دیگر لایه‌ها می‌باشد. توجه به این نکته حائز اهمیت است که در شکل ۳، هر دایره تجمیع‌شده عمل جمع و آستانه‌گذاری (عبور از تابع غیرخطی سیگموئید) است. در حقیقت هر دایره توپر در شکل ۳، مدلی است از جمع‌کننده و بلوک آستانه‌گذاری نشان‌داده شده در شکل ۲، که به منظور سهولت نمایش به این فرم نشان داده شده است. با توجه به شکل، خروجی عصب iام (در لایه آخر) را می‌توان به صورت زیر نشان داد:



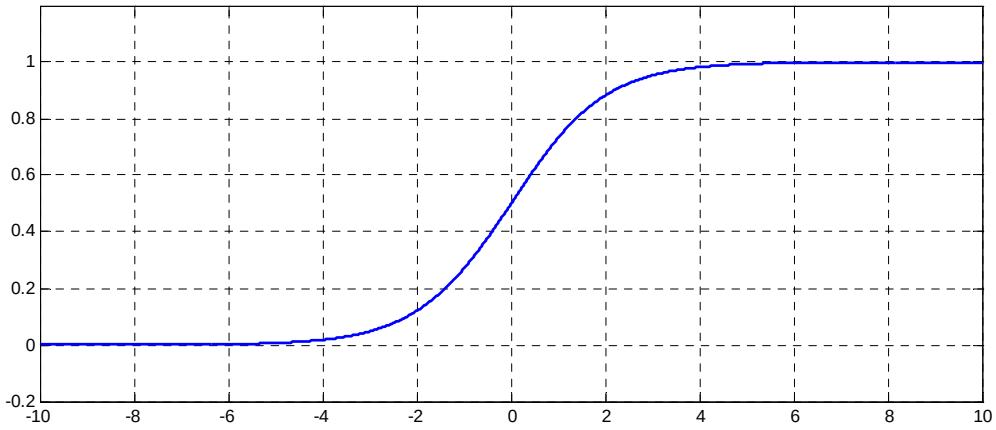
شکل ۳- پرسپترون ۳ لایه با اتصالات کامل.

$$O_i = sgm\left(\sum_m sgm\left(\sum_l x_i w_{lm}^h\right) w_{mi}^o\right) \quad (۱)$$

که در آن، h و o به ترتیب نشان‌دهنده لایه نهان و لایه خروجی بوده و منظور از w همان وزن‌های لایه‌ها می‌باشد. sgm نیز تابع سیگموئید است که به صورت زیر تعریف می‌گردد:

$$sgm(x) = \frac{1}{1+e^{-x}} \quad (۲)$$

شکل ۴ تابع سیگموئید را نشان می‌دهد. می‌توان دید که این تابع رفتاری هموار بین ۰ و ۱ دارد.



شکل ۴- رفتار تابع سیگموئید.

آموزش شبکه به روش پس‌انتشار خطا

به طور کلی شبکه‌های عصبی مصنوعی از لحاظ یادگیری بر دو دسته‌اند: شبکه‌های وزن‌ثابت و شبکه‌های با وزن متغیر (شبکه‌های یادگیرنده). خود شبکه های یادگیرنده نیز به دو دسته باسرپرست (Supervised) و بدون سرپرست (Unsupervised) تقسیم می‌شوند. در شبکه‌های باسرپرست، در فاز آموزش از نمونه‌هایی استفاده می‌گردد که خروجی ایده‌آل متناظر با آن‌ها از پیش دانسته است. به عبارت دیگر در این گونه شبکه‌ها، نمونه‌های داده ورودی، برچسب دارند. در شبکه‌های بی‌سرپرست، بر اساس یک معیار (مثلاً فاصله) و بر اساس نوعی رقابت، خروجی مورد نظر در کلاس جداگانه قرار می‌گیرد.

با توجه به این‌که شبکه عصبی، مدل ساده شده اعصاب بدن است، درست به مانند آن‌ها قابلیت یادگیری دارد. به عبارت دیگر، شبکه با استفاده از اطلاعاتی که از ورودی و توسط سرپرست خود دریافت می‌کند، قادر به فراگیری روند موجود در الگوهاست. لذا به طور مشابه با انسان، روند یادگیری در شبکه عصبی نیز از مدل‌های انسانی الهام گرفته است بدین صورت که مثال‌های بسیاری را به دفعات بایستی به شبکه ارائه نمود تا بتواند با تغییر وزن‌های شبکه، خروجی موردنظر را دنبال کند.

ارائه نمونه داده‌های ورودی به شبکه عصبی به دو روش امکان‌پذیر است:

۱. روش ارائه یک‌جا (Batch mode): در این روش، تمام نمونه‌ها به شبکه ارائه می‌گردند و در آخر، خطای شبکه نسبت به کل نمونه‌ها محاسبه گشته و وزن‌ها بر اساس آن خطا تغییر می‌کنند. در مرحله بعد، مجدداً تمام داده‌ها یکبار دیگر به شبکه ارائه شده و روند فوق نظیر به نظیر انجام می‌پذیرد تا نهایتاً خطا به سطح قابل قبولی برسد. مسلماً این روش پیچیده و زمان‌بر بوده و نیاز به حافظه زیادی دارد. همچنین امکان گیرکردن الگوریتم در می‌نیمم‌های محلی وجود دارد.

۲. روش ارائه الگو (Pattern mode): در این روش، در هربار نمونه‌ها به صورت تک‌تک به شبکه داده شده و خطای متناظر با همان داده بلافاصله محاسبه شده و بر اساس آن، وزن‌های شبکه تغییر می‌کنند. سپس نمونه بعدی به شبکه ارائه شده و روند بالا مشابهاً انجام می‌پذیرد. چون در این روش، در هر گام، اصلاح

وزن‌ها بر اساس هر نمونه انجام می پذیرد، الگوریتم همگرایی خوبی داشته و با توجه به ماهیت تصادفی موجود در ارائه تکی داده‌ها، خطر می‌نیمم‌های محلی منتفی است.

به منظور آموزش شبکه و اصلاح وزن‌ها تا رسیدن به یک خطای معنادار، روش‌های بسیار زیادی وجود دارد. یکی از مشهورترین این روش‌ها، الگوریتم پس‌انتشار خطا (Error back propagation algorithm) است که در ادامه توضیح داده می‌شود.

الگوریتم پس‌انتشار خطا:

این الگوریتم که در سال ۱۹۸۶ توسط روملهارت و مک‌کلیلاند پیشنهاد گردید، در شبکه‌های عصبی پیش‌سو (Feed forward) مورد استفاده قرار می‌گیرد. پیش‌سو بودن به این معناست که نرون‌های مصنوعی در لایه‌های متوالی قرار گرفته‌اند و خروجی (سیگنال) خود را رو به جلو می‌فرستند. واژه پس‌انتشار نیز به معنای این است که خطاها به سمت عقب در شبکه تغذیه می‌شوند تا وزن‌ها را اصلاح کنند و پس از آن، مجدداً ورودی مسیر پیش‌سوی خود تا خروجی را تکرار کند. روش پس‌انتشار خطا از روش‌های باسرپرست است به این مفهوم که نمونه‌های ورودی برچسب خورده‌اند و خروجی مورد انتظار هر یک از آن‌ها از پیش دانسته است. لذا خروجی شبکه با این خروجی‌های ایده‌آل مقایسه شده و خطای شبکه محاسبه می‌گردد. در این الگوریتم ابتدا فرض بر این است که وزن‌های شبکه به طور تصادفی انتخاب شده‌اند. در هر گام خروجی شبکه محاسبه شده و بر حسب میزان اختلاف آن با خروجی مطلوب، وزن‌ها تصحیح می‌گردند تا در نهایت این خطا، می‌نیمم شود. در الگوریتم پس‌انتشار خطا، تابع تحریک هر عصب به صورت جمع وزن‌دار ورودی‌های مربوط به آن عصب درنظر گرفته می‌شود. بدین ترتیب با فرض این‌که w وزن‌های متناظربین لایه ورودی و لایه بعد باشد می‌توان نوشت:

$$A_j(\bar{x},\bar{w}) = \sum_{i=0}^{n} x_i w_{ji} \qquad (۳)$$

به وضوح می‌توان دیدکه خروجی تابع تحریک عصب فقط به ورودی و وزن‌های متناظر بستگی دارد. با فرض این‌که تابع خروجی، سیگموئید باشد می توان خروجی عصب j ام را به صورت زیر نوشت:

$$O_j(\bar{x},\bar{w}) = sgm\left(A_j(\bar{x},\bar{w})\right) = \frac{1}{1+e^{A_j(\bar{x},\bar{w})}} \qquad (۴)$$

همانگونه که در شکل ۴ دیده شد، تابع سیگموئید به ازای اعداد منفی بزرگ، بسیار نزدیک به صفر است و برای اعداد مثبت بزرگ، مقداری بسیار نزدیک به ۱ دارد و در این بین به طور هموار تغییر می‌کند به نحوی که در $x=0$ دقیقاً از حد واسط بازه [۰,۱] یعنی ۰/۵ عبور می‌کند. همچنین با دقت در رابطه (۴) درمی‌یابیم که خروجی فقط به مقدار تابع تحریک بستگی دارد که به نوبه خود به ورودی و وزن‌ها مرتبط می‌شود. لذا برای تغییر خروجی باید وزن‌ها تغییر کنند. آنچنان‌که پیش از این نیز بیان شد، هدف فرآیند آموزش، رسیدن به خروجی مطلوب (یا نزدیک به مطلوب) است. بدین ترتیب ابتدا باید تابع خطای هر نرون را تعریف کنیم. این خطا از اختلاف خروجی واقعی شبکه و خروجی مورد انتظار به صورت زیر بدست می‌آید:

$$E_j(\bar{x},\bar{w},d_j)=\left(O_j(\bar{x},\bar{w})-d_j\right)^2 \qquad (۵)$$

انتخاب مربع تفاضل بین خروجی واقعی(O_j) و خروجی مطلوب (d_j) از چندین جنبه قابل بحث است؛ اولاً با استفاده از توان دوم، مقدار خطا همواره مثبت خواهد بود؛ ثانیاً اگر اختلاف بین خروجی واقعی و مطلوب زیاد باشد، توان دوم منجر به بزرگ‌تر شدن این عدد می‌شود و بالعکس اگر اختلاف بین خروجی واقعی و مطلوب کم باشد، توان دوم منجر به کوچک‌تر شدن آن می‌گردد. بر این اساس می‌توان خطای کلی شبکه را به فرم مجموع خطای تک‌تک عصب‌های لایه خروجی نوشت. لذا داریم:

$$E(\bar{x},\bar{w},\bar{d})=\sum_j E_j(\bar{x},\bar{w},d_j)=\sum_j\left(O_j(\bar{x},\bar{w})-d_j\right)^2 \qquad (۶)$$

حال بایستی به بررسی ارتباط خطا با ورودی‌ها، وزن‌ها و خروجی‌ها بپردازیم. برای این کار روش‌های متفاوتی وجود دارد که برخی از مهمترین آن ها عبارتند از:

۱. روش گرادیان شیب (Gradient descent)
۲. روش نیوتن
۳. روش اندازه حرکت (Momentum)
۴. روش آنتروپی متقابل (Cross entropy)
۵. روش Marquarlt-Levenberg

در این خودآموز، از روش گرادیان شیب به دلیل این‌که ساده‌ترین و درعین حال پرکاربردترین روش است، استفاده کرده‌ایم که در ادامه روابط آن بیان خواهد شد. در روش گرادیان شیب، ابتدا یک تابع هزینه درجه دوم تعریف می‌گردد که عبارت است از:

$$J(w)=\frac{1}{N}\sum_{i=1}^{N}J(w,i)=\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\sum_{j=1}^{L_o}\left(O_j-d_j\right)^2\right) \qquad (۷)$$

که در آن L_o بعد خروجی (تعداد نرون‌های لایه خروجی) است. هدف نهایی الگوریتم پس‌انتشار خطا، می‌نیمم کردن این تابع هزینه است. بر اساس روش گرادیان شیب، با توجه به درجه ۲ و مثبت بودن تابع هزینه، فرض می‌شود این تابع رفتاری سهموی دارد. لذا برای رسیدن به می‌نیمم کلی آن بایستی در خلاف جهت شیب تابع حرکت کنیم. بنابراین با فرض این‌که وزن‌ها در ابتدای کار به صورت تصادفی انتخاب شده باشند، باید شیب تابع خطا را نسبت به وزن‌ها محاسبه نموده و در جهت خلاف آن، وزن‌ها را تغییر دهیم و این روند را تا آن‌جا ادامه دهیم که به می‌نیمم کلی یا یک خطای قابل قبول برسیم. بنابراین میزان تغییر وزن‌ها در هرگام عبارت است از:

$$\Delta w_{ji}=-\eta\frac{\partial E}{\partial w_{ji}} \qquad (۸)$$

که در آن η ثابت اصلاح وزن‌هاست و توسط کاربر انتخاب می‌شود. این ثابت نرخ همگرایی الگوریتم را تعیین می‌کند. لذا به وضوح هر چه مقدار این ثابت بیشتر باشد، میزان تغییرات در هر گام بیشتر خواهد بود و بالعکس. طبق رابطه (۸) به منظور یافتن میزان اصلاح وزن‌ها در هر گام، باید مشتق خطا را برحسب وزن‌ها بدست آوریم. بدین منظور با استفاده از قاعده زنجیره‌ای می‌توان نوشت:

$$\frac{\partial E}{\partial w_{ji}}=\frac{\partial E}{\partial O_j}\cdot\frac{\partial O_j}{\partial w_{ji}} \qquad (۹)$$

با توجه به رابطه (۶) می‌توان نوشت:

$$\frac{\partial E}{\partial O_j}=2\left(O_j-d_j\right) \qquad (۱۰)$$

همچنین با استفاده مجدد از قاعده زنجیره‌ای و با توجه به روابط (۳) و (۴) می‌توان نوشت:

$$\frac{\partial O_j}{\partial w_{ji}} = \frac{\partial O_j}{\partial A_j} \cdot \frac{\partial A_j}{\partial w_{ji}} = \left(O_j\left(1-O_j\right)\right)\cdot x_i \qquad (۱۱)$$

در رابطه بالا برای محاسبه $\partial O_j / \partial A_j$ با توجه به رابطه (۳) و با استناد به خاصیت زیر برای تابع سیگموئید استفاده شده است که در آن می‌توان مشتق تابع سیگموئید را بر حسب خود تابع نوشت:

$$\frac{d}{dx} sgm(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} = \left(1-sgm(x)\right)sgm(x) \qquad (۱۲)$$

با قرار دادن (۱۰) و (۱۱) در رابطه (۹) خواهیم داشت:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial O_j} \cdot \frac{\partial O_j}{\partial w_{ji}} = 2(O_j - d_j)O_j(1-O_j)x_i \qquad (۱۳)$$

اکنون با جایگزینی (۱۳) در (۸)، رابطه نهایی میزان اصلاح وزن‌ها در هر گام از الگوریتم پس‌انتشار خطا به صورت زیر بدست می‌آید:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -2\eta(O_j - d_j)O_j(1-O_j)x_i \qquad (۱۴)$$

از رابطه بالا در اصلاح وزن‌های یک شبکه عصبی دو لایه (ورودی-خروجی) که فقط یک دسته وزن دارند می‌توان استفاده نمود. با فرض وجود لایه پنهان بایستی دو دسته وزن در هر گام از اجرای الگوریتم پس‌انتشار خطا، اصلاح شوند: وزن‌های اتصالات لایه ورودی به لایه پنهان ($\overline{v}$) و وزن‌های اتصالات لایه پنهان به لایه خروجی ($\overline{w}$). در این حالت، خطا علاوه بر وابستگی به وزن‌های اتصالات لایه پنهان به لایه خروجی یا همان w_{ji} ، به وزن‌های v_{ik} نیز وابسته است. مجدداً با استفاده از ایده گرادیان شیب می‌توان نوشت:

$$\Delta v_{ik} = -\eta \frac{\partial E}{\partial v_{ik}} = -\eta \frac{\partial E}{\partial x_i} \cdot \frac{\partial x_i}{\partial v_{ik}} = -\eta \frac{\partial E}{\partial x_i} \cdot \left(x_i\left(1-x_i\right)v_{ik}\right) \qquad (۱۵)$$

همچنین با توجه به این‌که در حالتی که شبکه عصبی دارای ۳ لایه است، ورودی لایه خروجی در حقیقت خروجی لایه پنهان است، رابطه (۱۳) به صورت زیر تغییر می‌یابد:

$$\frac{\partial E}{\partial w_{ji}} = 2(O_j - d_j)O_j(1-O_j)w_{ji} \qquad (۱۶)$$

اگر شبکه عصبی دارای تعداد لایه‌های بیشتری باشد، با استفاده از روندی مشابه با روند بالا می‌توان میزان اصلاح وزن های هر لایه را بدست آورد. در عمل می‌توان نشان داد که یک شبکه عصبی با ۳ لایه می‌تواند راندمانی مشابه با شبکه‌های با لایه های بیشتر داشته باشد، بنابراین از آنجا که افزایش تعداد لایه‌ها، الگوریتم یادگیری را پیچیده‌ترمی‌کند، مرسوم است که از شبکه‌های عصبی با ۳ لایه (ورودی، نهان و خروجی) استفاده گردد.

روند شبیه‌سازی مسائل

به منظور شبیه‌سازی یک طبقه‌بندی مسأله با استفاده از شبکه‌های عصبی به روش با سرپرست (با فرض در اختیار داشتن داده‌های برچسب دار) اولین کار انتخاب ابعاد شبکه است. با توجه به بحث پیشین، انتخاب تعداد لایه‌ها محدود به ۳ لایه می شود. در لایه ورودی باید به تعداد ابعاد هر الگوی ورودی، نرون قرار دهیم. بنابراین سایز لایه ورودی را بعد داده‌های ورودی تعیین می‌کند. در لایه خروجی نیز به وضوح باید به تعداد کلاس‌ها، نرون داشته باشیم. در حالت ایده‌آل، با آمدن ورودی مربوط به یک کلاس انتظار داریم نرون مربوط به آن کلاس مقدار۱ و مابقی نرون‌ها مقدار۰ را به خود بگیرند. اما در عمل با توجه به تابع سیگموئید مورد استفاده، قدار خروجی شبکه عددی بین صفر و یک است. لذا در حالت عملی نرونی از لایه خروجی که اصطلاحاً بیش از بقیه روشن شده باشد (مقدار آن بزرگ‌تر از بقیه باشد) کلاس مربوط به آن داده را مشخص می‌کند. در مورد تعداد نرون‌های لایه میانی، مبنای خاصی وجود ندارد و معمولاً با صحیح و خطا به نحوی انتخاب می‌گردد که شبکه جواب معقولی در اختیار بگذارد. باید دقت داشت که اگر شبکه خیلی پیچیده باشد، دقیقاً رفتار الگوهای ورودی را یاد خواهد گرفت و لذا اگر داده‌ای کمی نسبت به داده‌های آموزشی تغییر کند، شبکه به راحتی قادر به دنبال کردن آن نخواهد بود. در این حالت اصطلاحاً گفته می‌شود که شبکه عصبی قابلیت تعمیم ندارد. این چنین شبکه‌هایی را Over-loaded networks گویند.

قدم بعدی در شبیه‌سازی، انتخاب نوع تابع خروجی نرون است که در شبکه‌های MLP که با الگوریتم پس‌انتشار خطا کار می‌کنند، عمدتاً تابع سیگموئید استفاده می‌شود. دلیل این امر هم مشتق‌گیری ساده و ارتباط مستقیم مشتق تابع با خود تابع است (رابطه (۱۲)).

پیش از شروع شبیه‌سازی، باید داده‌های ورودی را به دو گروه تقسیم نمود:

۱. داده‌های آموزش: این داده‌ها از میان داده‌های برچسب‌دار و به منظور آموزش شبکه به کار می‌روند. عمدتاً از میان کل داده‌ها ٪۶۰ تا ٪۷۰ آن‌ها را (به طور تصادفی یا با یک پیش‌فرض) به عنوان داده‌های آموزش انتخاب می‌کنند. پس از آن‌که شبکه توسط این داده‌ها آموزش دید، وزن‌ها مقدار نهایی خود را یافته‌اند به نحوی که شبکه برای داده های آموزش، کمترین خطا را بدست می‌دهد.

۲. داده‌های تست: پس از آن‌که شبکه توسط داده‌های آموزش تا رسیدن به حداقل خطا آموزش یافت، مابقی داده‌ها (٪۴۰ تا ٪۳۰ باقی‌مانده) که در آموزش نقشی نداشته‌اند به عنوان ورودی به شبکه داده شده و پاسخ شبکه با پاسخ مطلوب (برچسب آن‌ها) مقایسه می‌گردد و بدین ترتیب راندمان شبکه آموزش‌دیده محک زده می‌شود.

شایان ذکر است که اگر تعداد نمونه‌های آموزش به حد کافی زیاد باشد، استفاده از روش ارائه الگو جواب‌های بهتری می‌دهد اما در کاربردهایی که تعداد نمونه‌های برچسب دار آموزش کم باشند، هر دو روش ارائه الگو و ارائه یک‌جا به یک جواب یکسان منجر خواهند شد. همچنین باید توجه داشت که در اولین گام اجرای الگوریتم، وزن تمامی لایه‌ها به طور تصادفی انتخاب می‌شوند و در هر گام با استفاده از روش پس‌انتشار خطا، وزن‌ها تصحیح می‌شوند. اشاره به این نکته حائز اهمیت است که پس از آن‌که تمام نمونه‌ها یک‌بار به شبکه ارائه شدند، در بار بعدی (اپوک بعد) ابتدا داده‌های آموزشی به طور تصادفی بر زده شوند. این کار به ویژه سبب می‌شود که شبکه عصبی به ازای نمونه‌های خاص بایاس نشود و همچنین از گیر کردن در می نیمم‌های محلی جلوگیری می‌کند.

پس از اجرای الگوریتم تصحیح وزن‌ها تا رسیدن به می‌نیمم خطای کلی، چندین معیار برای توقف الگوریتم پیشنهاد می‌گردد که مهمترین آن‌ها عبارتند از:

۱. اگر خطا (تابع هزینه) که قرار است می‌نیمم شود، از یک سطح آستانه کمتر شود، می‌توان پذیرفت که شبکه با خطای قابل قبولی آموزش دیده است. این شرط چنین بیان می‌شود:

$$J(w) < \varepsilon_0 \qquad (۱۷)$$

۲. اگر میزان تغییرات خطا در طی دو گام متوالی کمتر از یک سطح آستانه باشد یا به عبارت دیگر روند کاهش خطا با سرعت کند انجام شود، می‌توان چنین برداشت کرد که به حوالی می‌نیمم کلی رسیده‌ایم. لذا در این حالت نیز تصمیم به توقف الگوریتم آموزش گرفته می‌شود. بیان ریاضی این شرط نیز چنین است:

$$\left|\nabla_w J(w)\right| < \varepsilon_0 \qquad (۱۸)$$

«والّذین جاهدوا فینا لنهدینّهم سبلنا»
با آرزوی موفقیت
امید صیّادی