استاد: فاطمه منصوری | نیمسال دوم ۱۴۰۲-۱۴۰۳

شبکه‌های عصبی

یادگیری عمیق

گردآورنده: سینا جعفری

۱ مقدمه

شبکه‌های عصبی یکی از شاخه‌های مهم و پرکاربرد حوزه یادگیری نظارت‌شده و روش‌های محاسباتی نوین تلقی می‌شوند. این شاخه، با بهره‌گیری از ساختار عصبی مغز انسان‌ها طراحی شده و می‌تواند تقریب و راهکارهای بسیار مناسبی برای حل مسائل مختلف نظارت‌شده در یادگیری ماشین ارائه دهد. شباهت این روش به ساختار عصبی انسان را می‌توان از دو جهت زیر بررسی کرد.

۱. ساختار عصبی انسان همانند ساختار مدل‌های شبکه عصبی، متناسب با یک مجموعه اطلاعات آموزش می‌بیند.

۲. هر دو ساختار، براساس یک سیستم وزن‌دهی از پارامترها کار می‌کنند.

یکی از ساده‌ترین انواع شبکه‌های عصبی مصنوعی[۱]، پرسپترون‌ها[۲] هستند که توسط وارن مک‌کالک[۳] و والتر پیتس[۴] معرفی شدند [۱]. اما این الگوریتم اولین‌بار در آزمایشگاه کرنل آرونتیکال توسط فرانک روزنبلات[5] پیاده‌سازی شد. این الگوریتم در دو نوع تک لایه و چندلایه کار می‌کند که در ادامه فصل به‌طور کامل به هر دوی آن خواهیم پرداخت.

۱.۱ اجزای شبکه عصبی

شبکه‌های عصبی به نوع گسترده‌ای از مدل‌های غیرخطی اشاره می‌کنند که شامل ترکیبی از ضرب‌های ماتریسی و سایر عملیات غیرخطی بر روی ورودی هستند. بطور کلی شبکه عصبی از چهار بخش اصلی **ورودی**، **وزن‌ها (پارامترهای قابل‌یادگیری مدل**[۶]**)**، **لایه مخفی**[۷] و **خروجی** تشکیل می‌شود که در ادامه هرکدام را شرح خواهیم داد.

ورودی. ورودی شبکه‌های عصبی باید ماتریسی از اعداد باشند. اگر مجموعه داده S را در اختیار داشته باشیم که:

$$S = ((x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)) \qquad s.t. \qquad x_i \in \mathbb{R}^d, \quad y_i \in \mathbb{R}$$

که در این‌جا m و d به ترتیب تعداد نمونه‌ها و اندازه ابعاد فضای ویژگی هستند، می‌توانیم یک فضای ورودی از نوع ماتریس تعریف کنیم که فقط شامل x_iها می‌باشد. بنابراین داریم:

$$X = (x_1, x_2, \dots, x_m) \qquad s.t. \qquad x_i \in \mathbb{R}^d$$

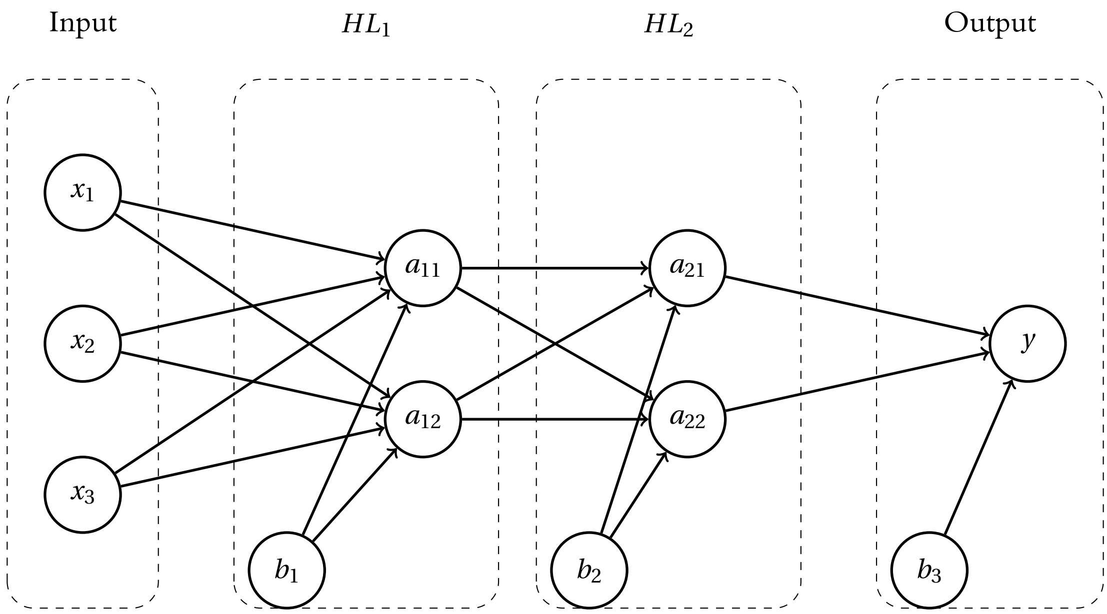
[1] Artificial Neural Network
[2] Perceptron
[3] Warren
[4] Pitts
[5] Frank Rosenblatt
[6] Trainable Parameters
[7] Hidden Layer

توجه کنید در مسائلی که مجموعه داده‌ها همگی تصویر هستند، هر نمونه (تصویر) یک ماتریس با ابعاد $w \times h \times c$ می‌باشد. یعنی در این حالت فضای ویژگی برابرست با:

$$X = (x_1, x_2, \dots, x_m) \qquad s.t. \qquad x_i \in \mathbb{R}^{w \times h \times c}$$

که در این‌جا w، h و c به ترتیب برابرند با طول، ارتفاع و تعداد کانال‌های هر تصویر.

لایه مخفی. یکی از مهم‌ترین اجزای سازنده هر شبکه عصبی، لایه‌های مخفی هستند. تعداد این لایه‌ها و نوع معماری که در آن‌ها قرار دارند، تأثیر بسیار مستقیمی در عملکرد یک شبکه عصبی می‌توانند داشته باشند. هر لایه مخفی در شبکه عصبی، مانند یک پشته[۸] دارای تعدادی نورون[۹] است که مقدار هرکدام از آن‌ها یک ابرپارامتر محسوب می‌شود و نمی‌توان به سادگی در مورد تعداد آن برای یک مسئله خاص نظر داد.



شکل ۱: یک شبکه با چند لایه مخفی.

به عنوان مثال، در شکل ۱ یک شبکه عصبی با چند لایه مخفی (در اصطلاح به این نوع شبکه‌ها، شبکه عصبی چندلایه یا پرسپترون چند لایه که به اختصار MLP [۱۰] است، گوییم) داریم که دارای دو لایه مخفی است. همان‌طور که مشاهده می‌کنید، هر لایه مخفی دارای سه نورون است که هر نورون برابرست با حاصل‌ضرب نورون‌های لایه قبل در وزن‌های مربوطه.

وزن‌ها. هر لایه در شبکه عصبی، توسط وزن‌ها به یکدیگر متصل می‌شوند. این وزن‌ها دقیقاً همان پارامترهای قابل یادگیری مدل شبکه عصبی هستند که در طول فرآیند یادگیری، باید بروزرسانی شوند تا به مقدار بهینه خود نزدیک شوند. معمولاً مقداردهی اولیه این وزن‌ها، به صورت تصادفی و از توزیع نرمال انتخاب می‌شود، اما روش‌های بهینه‌تر و کاراتری نیز برای مقداردهی اولیه وزن‌ها وجود دارد. یکی از این روش‌های مؤثر که اخیراً نیز در خیلی از شبکه‌های عصبی استفاده شده است، استفاده از مدل‌های از پیش آموزش دیده[۱۱] است، که در این روش از وزن‌های یک شبکه عصبی آموزش دیده به عنوان مقادیر اولیه وزن‌های شبکه عصبی موردنظر استفاده می‌شود. بنابراین در شکل ۱، بین لایه ورودی و لایه مخفی اول ماتریس وزن اول، بین لایه مخفی اول و دوم ماتریس وزن دوم و بین لایه مخفی دوم و خروجی ماتریس وزن سوم قرار دارند که به ترتیب با نمادهای $W^{[1]}$، $W^{[2]}$ و $W^{[3]}$ آن‌ها را نمایش می‌دهیم. ماتریس وزن کلی یک شبکه عصبی را W نمایش داده و به صورت زیر تعریف می‌کنیم:

$$W = \left[W^{[1]}, \quad W^{[2]}, \quad W^{[3]} \right]$$

[8]Stack
[9]Neuron
[10]Multi-Layer Perceptron
[11]Pre-trained Models

خروجی. لایه خروجی هر شبکه عصبی، معمولاً دو حالت مختلف دارد که متناسب است با نوع مسئله. اگر مسئله رگرسیون باشد، در لایه آخر یا خروجی تنها یک نورون وجود دارد که مقدار آن برابرست با مقدار پیش‌بینی شده توسط مدل. اما اگر مسئله از نوع دسته‌بندی باشد، در لایه خروجی به تعداد دسته‌های مسئله نورون خواهیم داشت که هرکدام یک مقدار خاص دارند. اما در این حالت چگونه می‌توان دسته پیش‌بینی شده توسط مدل را بدست آورد؟ اعدادی که هر نورون در لایه آخر دارند ممکن است از نظر عددی با هم اختلاف داشته باشند و نمی‌توان آن‌ها را در این مقیاس مقایسه کرد. یکی از راه‌هایی که برای حل این مشکل ارائه شده است، استفاده از لایه softmax می‌باشد. تابع softmax یک ورودی بردار دریافت کرده و همه اعداد درون بردار را به مقیاس ۰ تا ۱ می‌برد، به‌طوری که مجموع همه اعضای بردار برابر شود با ۱. به عبارت دیگر، مقدار تمامی اعضای بردار خروجی، تابعی از احتمال می‌شوند. این تابع به صورت زیر تعریف می‌شود:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{۱}$$

که در این‌جا، K برابر با طول بردار می‌باشد. به عنوان مثال، اگر بردار $\vec{z}$ برابر باشد با:

$$\vec{z} = \begin{bmatrix}12, 15, 17, 10\end{bmatrix}$$

مقدار تابع softmax بر روی این بردار برابرست با:

$$\sigma(\vec{z}) = \begin{bmatrix}0.0059, \ 0.1184, \ 0.8749, \ 0.0008\end{bmatrix}$$

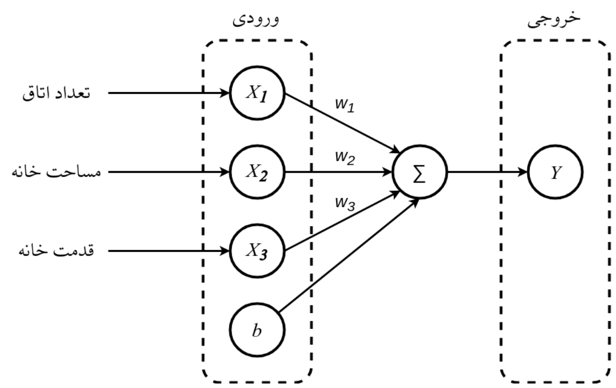
که مجموع اعضای بردار بدست آمده برابرست با ۱.

۲ پرسپترون تک لایه

در این‌جا برای سادگی، یک شبکه عصبی با ورودی x را با نماد $\bar{h}_\theta(x)$ نمایش می‌دهیم. بنابراین برای تابع $\bar{h}_\theta$ داریم:

$$\bar{h}_\theta : X \to Y \tag{۲}$$

یعنی تابع $\bar{h}_\theta$ یک نگاشت از فضای ویژگی به فضای برچسب است. حال فرض کنید مجموعه داده‌ای با ویژگی‌های قد، سن و وزن افراد در اختیار داریم و قرار است با این سه ویژگی، دیابت داشتن یا نداشتن یک شخص را پیش‌بینی کنیم، یعنی $y \in \{0, 1\}$. روال کار یک پرسپترون تک لایه بسیار ساده می‌باشد و مقدار هر ویژگی در یک پارامتر قابل یادگیری ضرب و سپس همه‌ی مقادیر با هم جمع می‌شوند.



شکل ۲: یک شبکه پرسپترون تک لایه.

سپس بعد از جمع کردن تمام مقادیر، یک تابع روی خروجی آن اعمال می‌شود که به آن تابع فعالساز[12] گوییم. این توابع اغلب غیرخطی هستند و درواقع این توابع غیرخطی در شبکه عصبی باعث غیرخطی شدن عملکرد آن‌ها می‌شود. برای مثال یکی از معروف‌ترین توابع

[12] Actication Function

فعالساز، تابع $ReLU$[۱۳] است که به صورت زیر تعریف می‌شود:

$$ReLU(x) = x^+ = max(0,x) = \frac{x+|x|}{2} = \begin{cases} x & \text{if } x > 0 \\ 0 & o.w. \end{cases} \quad (۳)$$

اما در پرسپترون تک لایه این تابع، خطی و نام آن تابع پله[۱۴] است، که به‌صورت زیر تعریف می‌شود:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & o.w. \end{cases} \quad (۴)$$

بنابراین، حاصل یک شبکه پرسپترون تک لایه (شکل ۲) برابرست با:

$$y = f(w_1x_1 + w_2x_2 + w_3x_3 + b) = f(\sum_{i=1}^{3} w_ix_i + b) \quad (۵)$$

که در این رابطه، w_i وزن پارامتر iام، x_i مقدار ویژگی iام و در آخر b نیز مقدار بایاس[۱۵] است. در شبکه‌های عصبی، بایاس به مقدار ثابتی گفته می‌شود که درون هر لایه از یک شبکه قرار دارد و با اضافه شدن به مجموع حاصل‌ضرب پارامترها در ویژگی‌ها، می‌تواند نقش اساسی‌ای در تعیین مقدار برچسب داشته باشد. البته در بسیاری از معماری‌ها و لایه‌های مختلف یک شبکه عصبی مقدار بایاس می‌تواند گذاشته نشود، اما در مدل‌های پایه شبکه عصبی این مقدار ثابت گذاشته می‌شود. در ادامه، همانند رگرسیون که می‌توانستیم ویژگی‌ها و پارامترهای مدل را درون دو بردار مجزا قرار دهیم، در شبکه‌های عصبی نیز می‌توانیم این کار را انجام دهیم. درواقع با انجام این کار عملیات ضرب و جمع بین دو عدد تبدیل به ضرب ماتریسی شده و برای حل حالات پیچیده‌تری که در ادامه ذکر می‌شود، محاسبات را ساده‌تر می‌کند. بنابراین اگر بردارهای W و X به ترتیب وزن‌های پارامتر مدل و مقادیر ویژگی‌ها باشند، داریم:

$$y = f(W^TX + b) \quad s.t. \quad X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}. \quad (۶)$$

[13] Rectified Linear Unit
[14] Step Function or Binary Step Function
[15] Bias

۳ توابع فعالساز

می‌توان از توابع غیرخطی معروف دیگری به‌جای تابع $ReLU$ که در بخش قبل آورده شد، استفاده کرد. همه این توابع باید یک نگاشت از فضای اعداد حقیقی به اعداد حقیقی باشند، یعنی:

$$\sigma : \mathbb{R} \to \mathbb{R} \qquad (۷)$$

برخی از این توابع غیرخطی عبارتنداز:

$$\sigma(z) = \max\left\{\alpha z,\ z\right\} = \begin{cases} z & \text{if } z > 0 \\ \alpha z & o.w. \end{cases} \qquad s.t. \qquad \alpha \in \left(0,\ 1\right) \qquad (\text{leaky-ReLU}) \qquad (۸)$$

$$\sigma(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & o.w. \end{cases} \qquad s.t. \qquad \alpha > 0 \qquad (\text{ELU})\ [2] \qquad (۹)$$

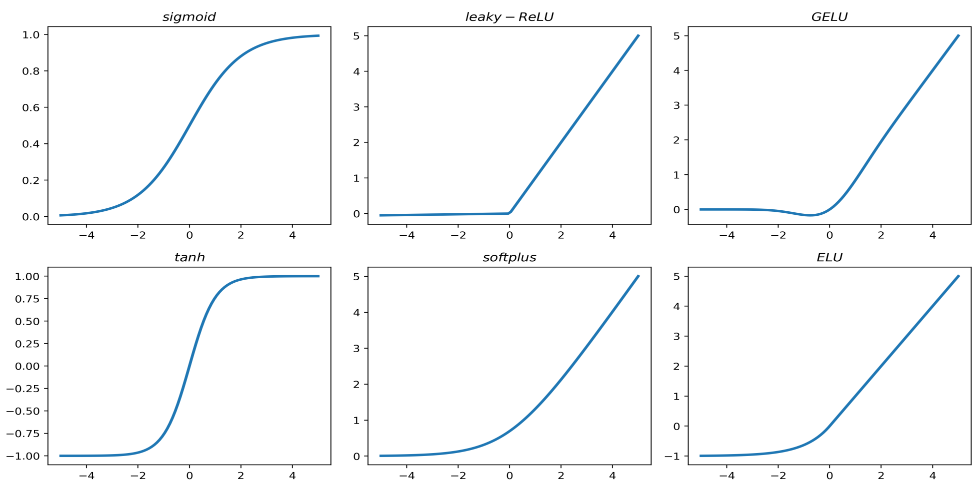
$$\sigma(z) = \frac{z}{2}\left[1 + \text{erf}(\frac{z}{\sqrt{2}})\right] \qquad (\text{GELU})\ [3] \qquad (۱۰)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad (\text{sigmoid}) \qquad (۱۱)$$

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \qquad (\tanh) \qquad (۱۲)$$

$$\sigma(z) = \frac{1}{\beta}\log(1 + e^{\beta z}) \qquad s.t. \qquad \beta > 0 \qquad (\text{softplus})\ [4] \qquad (۱۳)$$

در ادامه نمودار هرکدام از این توابع در شکل ۳ آورده شده است.



شکل ۳: نمودار توابع فعالساز.

یکی از ویژگی‌های مهمی که توابع فعالساز غیرخطی باید دارا باشند، **مشتق پذیر** بودن است. همان‌طور که می‌دانید در روش گرادیان کاهشی، مدام باید از عمل مشتق‌گیری برای یافتن جواب بهینه استفاده کرد. بنابراین مشتق پذیر بودن این توابع از اهمیت زیادی برخوردار هستند. اگر تابع فعالسازی که انتخاب می‌کنیم، در بعضی نقاط مشتق پذیر نباشد، آنگاه موجب صفر شدن بعضی ضرایب می‌شود که این اتفاق باعث توقف یادگیری می‌شود. در ادامه مشتق برخی توابع فعالساز ذکر شده را یادآوری می‌کنیم.

$$\frac{d}{dz}\sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ \alpha & o.w. \end{cases} \qquad s.t. \qquad \alpha \in (0,\ 1) \qquad \text{(derivative of leaky-ReLU)} \tag{۱۴}$$

$$\frac{d}{dz}\sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ \alpha e^{z} & o.w. \end{cases} \qquad s.t. \qquad \alpha > 0 \qquad \text{(derivative of ELU)} \tag{۱۵}$$

$$\frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z)) \qquad \text{(derivative of sigmoid)} \tag{۱۶}$$

$$\frac{d}{dz}\sigma(z) = 1-\sigma(z)^2 \qquad \text{(derivative of tanh)} \tag{۱۷}$$

$$\frac{d}{dz}\sigma(z) = \frac{1}{1+e^{-z}} \qquad \text{(derivative of softplus)} \tag{۱۸}$$

برای بررسی دقیق‌تر مشتق تابع $GELU$ که در رابطه ۱۰ قرار دارد، می‌توانید مقاله [۳] را بررسی کنید.

۴ الگوریتم پیش‌خور

فرآیند یادگیری در شبکه‌های عصبی طی دو مرحله رفت و برگشت صورت می‌گیرد که به ترتیب توسط الگوریتم‌های **پیش‌خور**[۱۶] و **پس‌انتشار**[۱۷] صورت می‌گیرد. در الگوریتم پیش‌خور تنها کافی است ورودی را به ترتیب در لایه‌های شبکه عصبی ضرب کرده و نتیجه حاصل را بدست آورد. در این مرحله، هیچ‌گونه بروزرسانی در پارامترهای شبکه صورت نمی‌گیرد و تنها خروجی و سپس مقدار تابع هزینه را می‌توان بدست آورد. در مرحله بعد، با استفاده از الگوریتم پس‌انتشار مشتق تابع هزینه را نسبت به تمامی پارامترهای شبکه محاسبه کرده و مقدار حاصل را جایگزین می‌کنیم. این روش در بخش‌های بعدی به‌طور کامل شرح داده خواهد شد.

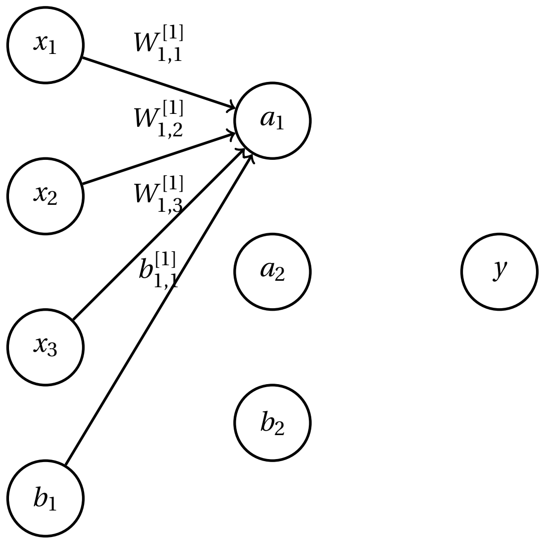
۵ پرسپترون چند لایه

اگر در مجموعه داده موردنظر، الگوهای خطی وجود نداشته باشد، آنگاه دیگر مدل‌های خطی نظیر پرسپترون تک لایه و رگرسیون خطی دیگر جوابگو نیستند. اضافه کردن لایه‌های مخفی متعدد به همراه توابع فعالساز غیرخطی، موجب ساخته شدن مدل‌های غیرخطی می‌شود که توانایی پیدا کردن الگوهای غیرخطی را دارند. می‌دانیم که هر لایه مخفی تعدادی وزن در ورودی دریافت و تعدادی وزن نیز در خروجی برمی‌گرداند. حال یک مثال از یک شبکه پرسپترون تک لایه به همراه ۳ نورون را بررسی می‌کنیم. در شکل ۴، مقدار نورون

[16] Feed-Forward
[17] Backpropagation

اول در لایه اول یعنی a_1 برابرست با:

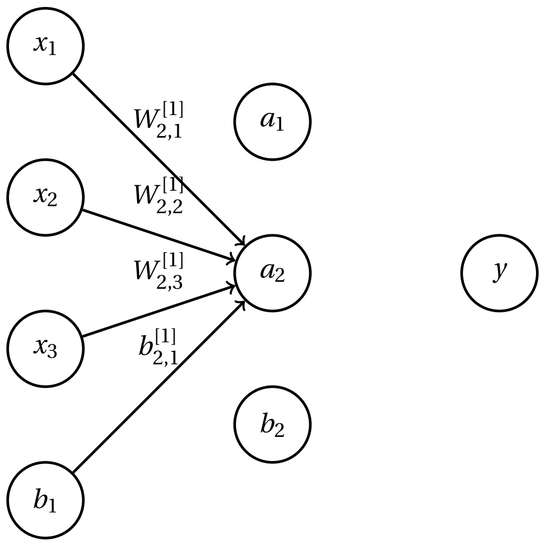
$$\begin{aligned} a_1 &= f(W^{[1]}_{1,1}x_1 + W^{[1]}_{1,2}x_2 + W^{[1]}_{1,3}x_3 + b^{[1]}_{1,1}) \\ &= f(\sum_{i=1}^{3} W^{[1]}_{1,i}x_i + b^{[1]}_{1,1}) \end{aligned} \tag{۱۹}$$



شکل ۴: یک شبکه پرسپترون با یک لایه مخفی به همراه سه نورون.

همچنین در شکل ۵ مقدار a_2 همانند حالت قبل برابرست با:

$$a_2 = f(\sum_{i=1}^{3} W^{[1]}_{2,i}x_i + b^{[1]}_{2,1}) \tag{۲۰}$$



شکل ۵: یک شبکه پرسپترون با یک لایه مخفی به همراه سه نورون.

که در این‌جا، $W^{[1]}$ نماد وزن لایه اول و نماد $W^{[1]}_{1,i}$ به معنی iـامین وزن از لایه اول که به نورون اول لایه بعد متصل می‌شود است. همچنین f نیز تابع فعالساز شبکه است که در این‌جا برای سادگی فرض کرده‌ایم تمام لایه‌ها دارای یک تابع فعالساز هستند. حال می‌توانیم مقادیر وزن‌ها را به‌صورت مجزا درون یک بردار قرار دهیم. پس بردارهای زیر را این‌گونه تعریف می‌کنیم:

$$
\begin{aligned}
W_1^{[1]} &= \left[W_{1,1}^{[1]}, \quad W_{1,2}^{[1]}, \quad W_{1,3}^{[1]}\right] \\
W_2^{[1]} &= \left[W_{2,1}^{[1]}, \quad W_{2,2}^{[1]}, \quad W_{3,3}^{[1]}\right]
\end{aligned} \tag{۲۱}
$$

پس می‌توانیم مقادیر a_1 و a_2 را به‌صورت ضرب دو بردار بنویسیم. داریم:

$$
\begin{aligned}
a_1 &= f(W_1^{[1]^T} X + b_{1,1}^{[1]}) \\
a_2 &= f(W_2^{[1]^T} X + b_{2,1}^{[1]})
\end{aligned} \tag{۲۲}
$$

حال می‌توانیم مقادیر a_1 و a_2 را درون برداری به نام h_1 قرار دهیم که به معنای مقدار لایه مخفی اول است. بنابراین داریم:

$$
h_1 = \left[a_1, \quad a_2\right] \tag{۲۳}
$$

در ادامه می‌توانیم دو برداری که در رابطه ۲۱ تعریف شده‌اند را درون یک بردار دیگر به نام $W^{[1]}$ و همچنین می‌توان دو مقدار بایاس لایه اول یعنی $b_{2,1}^{[1]}$ و $b_{1,1}^{[1]}$ را درون یک بردار به نام $b^{[1]}$ قرار داد. پس داریم:

$$
\begin{aligned}
W^{[1]} &= \left[W_1^{[1]}, \quad W_2^{[1]}\right] \\
b^{[1]} &= \left[b_{1,1}^{[1]}, \quad W_{2,1}^{[1]}\right]
\end{aligned} \tag{۲۴}
$$

حال رابطه ۲۳ را به‌صورت زیر بازنویسی می‌کنیم:

$$
h_1 = W^{[1]^T} X + b^{[1]} \tag{۲۵}
$$

اگر یک لایه مخفی دیگر به شکل ۵ اضافه کنیم، رابطه این لایه با استنتاج از حالت قبل، برابرست با:

$$
h_2 = W^{[2]^T} h_1 + b^{[2]} \tag{۲۶}
$$

بنابراین می‌توان نتیجه گرفت در لایه iـام از یک شبکه پرسپترون چند لایه، مقدار این لایه برابرست با:

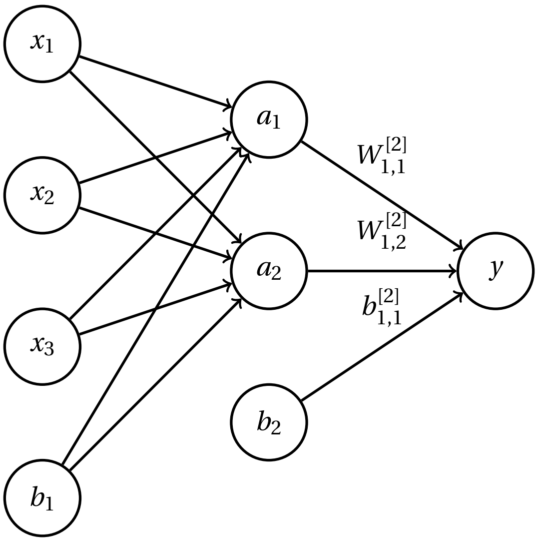
$$
h_i = W^{[i]^T} h_{i-1} + b^{[i]} \tag{۲۷}
$$

پس خروجی یک شبکه پرسپترون با r لایه، که با نماد $\bar{h}_\theta$ نمایش می‌دهیم، برابرست با:

$$
\bar{h}_\theta(X) = W^{[r]^T} h_{r-1} + b^{[r]} \tag{۲۸}
$$

بنابراین مقدار y در شکل ۶ برابرست با:

$$
\begin{aligned}
y &= a_1 W_{1,1}^{[2]} + a_2 W_{1,2}^{[2]} + b_{1,1}^{[2]} \\
&= W^{[2]^T} h_1 + b^{[2]}
\end{aligned} \tag{۲۹}
$$



شکل ۶: یک شبکه پرسپترون با یک لایه مخفی به همراه سه نورون.

مراجع

[1] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943.

[2] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[3] D. H. andKevin Gimpel, "Bridging nonlinearities and stochastic regularizers with gaussian error linear units," *CoRR*, vol. abs/1606.08415, 2016.

[4] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.