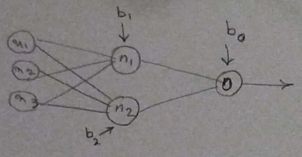


$$z = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

$$\{ w_{ij} = 1 \mid i = 1,2,3 \ , \ j = 1,2 \}$$

n_3	n_2	n_3	n_1	n_2
0	0	0	off	off
0	0	1	off	off
0	1	0	off	off
0	1	1	0.5	off
1	0	0	off	off
1	0	1	0.5	off
1	1	0	0.5	off
1	1	1	1.5	0.5

فرض: تنها یک ورودی مقدار ۱ باشد $\Rightarrow$ $z < 0$ شدن off

شرط $\quad 1 \times 1 + 0(1) + 0(1) + b_1 = 0 \implies b_1 < -1$

حداکثر مقدار $b_1 = -1.5$ در نظر گرفته

برای n_2 فرض می‌کنیم تنها دو ورودی ۱ باشند تا نزدن off

می‌شود یعنی $z < 0$

$(1)(1) + (1)(1) + (0)(1) + b_2 < 0$

حداکثر مقدار $b_2 = -2.5$ $\quad 2 + b_2 < 0 \implies$

n_1	n_2	n_3	$Z(n_1)$	$Z(n_2)$	Y
0	1	1	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	1	0

حداکثر فرض $\quad w_{10} + b_0 \geq 0 \longrightarrow w_{10} = 1 \ , \ b_0 \geq -1$

$w_{10} + w_{20} + b_0 < 0$

$\longrightarrow 1 + w_{20} - 1 < 0 \longrightarrow w_{20} < 0$

$w_{20} = -1$

$b_0 = -1$

$w_{10} = 1$

$\{ w_{ij} = 1 \text{ where } i = 1,2,3 \ , \ j = 1,2$

$w_{10} = 1 \ , \ w_{20} = -1$

$b_1 = -1.5$

$b_2 = -2.5$

$b_0 = -1$

Approach 1 Simple Logestic Regression (one neuron)

$$\hat{y} = \sigma(W_l x + b_l)$$

Approach 2 Simple softmax regression (Two neurons)

$$\hat{y} = \text{Softmax}(W_s x + b_s) = [\hat{y}_1, \hat{y}_2]^T$$

بطور کلی پیچیدگی بیشتر یک شبکه عصبی دلیلی بر Performance بیشتر نمی‌باشد و شبکه‌های softmax اصولاً جهت مدل‌های multi-classification طراحی گردیده‌اند. در این مثال باینری کلاسیفیکیشن داریم و یک نورون خروجی به تنهایی کارآمدی لازم را دارد. علاوه رنج هر دو تابع Approach 1,2 عددی بین [0,1] می‌باشد پس Approach 2 ظاهراً هیچ ارجحیتی در این مثال به Approach 1 ندارد

پس باید هر در این مسئله بتوان ارتباط بین (w_s, b_s) و (w_l, b_l) پیدا کرد

در اینصورت کارایی هر دو یکی است در حالیکه پیچیدگی Approach 2 بیشتر است.

در Approach 2 نورون دوم خروجی را تعیین می‌کند

معادله خروجی در این نورون ←

$$S(W_s x + b_s) = \frac{e^{w_{s2}x + b_{s2}}}{e^{w_{s2}x + b_{s2}} + e^{w_{s1}x + b_{s1}}}$$

صورت و مخرج را بر $e^{w_{s2}x + b_{s2}}$ تقسیم می‌کنیم

$$\frac{1}{1 + e^{w_{s1}x + b_{s1} - w_{s2}x - b_{s2}}}$$

$$-(w_l x + b_l) = -[(w_{s2} - w_{s1})x + (b_{s2} - b_{s1})]$$

معادله خروجی Approach 1

$$\frac{1}{1 + e^{-(w_l x + b_l)}}$$

نهایتاً داریم

$$(w_l, b_l) = (w_{s2} - w_{s1},\ b_{s2} - b_{s1})$$

تمرین 2

۱) در مسأله multi-class classification به دلایل زیر تابع Cross entropy مناسب می‌باشد

$$L(y-\hat{y}) = -\frac{1}{10} \sum_{i=1}^{10} (y_i - \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i))$$

در این تابع ما داریم:

$$L(y_i-\hat{y}_i) = \begin{cases} \log \hat{y}_i & \text{if } y_i = 1 \\ \log(1-\hat{y}_i) & \text{if } y_i = 0 \end{cases}$$

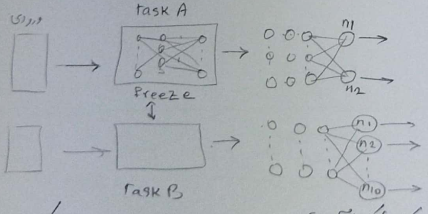
یعنی به ازای هر نمونه ورودی خطا در هر ۱۰ گره خروجی محاسبه می‌گردد هم خروجی ای که می باست ۱ باشد و هم بقیه خروجی ها که می باست صفر باشند.

قسمت بعد: می‌خواهیم از یک شبکه binary-class classification به یک شبکه Multi-class classification Transfer learning انجام دهیم چه پارامترهای جدیدی را باید tune کنیم؟

~~[illegible] small learning rate [illegible] epoch [illegible]~~

1- بعد از اینکه تعداد دلخواهی از لایه های میانی را Freeze کردیم می باست تغییراتی در output انجام دهیم و 2 خروجی مدل قبل را به ۱۰ خروجی تبدیل نمائیم که Fully connected می باشد

بنابراین در لایه آخر از $n\times 2 + n$ پارامتر تبدیل به $n\times 10 + n$ پارامتر می‌گردد



2- با تعداد epoch های کمتر از مرحله قبل کل شبکه را آموزش می دهیم اما وزنها در ابتدا همان وزنهای شبکه اولیه می باشند

3- حال قسمت Freeze شده را جدا کرده ورودی چند لایه آخر بعلاوه لایه output را اصطلاحاً وزنها را tune می کنیم

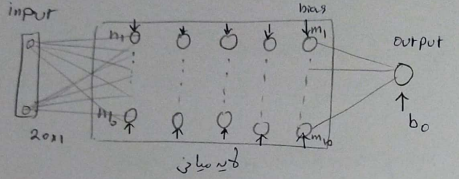
قسمت سوم از سومین سؤال

transfer learning همگی است باعث افزایش weight sharing یا مانی variance یا bias گردد

برای مثال اگر تعداد لایه‌های Freeze زیاد باشد و دیتاست جدید هم داده‌های زیادی داشته باشد ممکن است بتوانند لایه‌های آزاد به خوبی train شوند و high-bias اتفاق بیافتد

همچنین اگر دیتاست داده‌های کمی داشته باشد و تعداد لایه‌های Freeze کم باشند شبکه جدید با همین داده‌های دیتاست جدید adapt می‌شود و high-variance اتفاق می‌افتد بنابراین انتخاب تعداد لایه‌های Freeze بسیار مهم است.

قسمت چهارم از سومین سؤال



تعداد پارامترهای ورودی به لایه اول میانی $20 \times 10 + 10$

$+$

لایه‌های میانی $4\,(10 \times 10 + 10)$

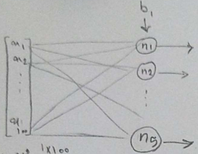
$+$

لایه خروجی $10 \times 1 + 1$

$= 661$

C)

Method 1:



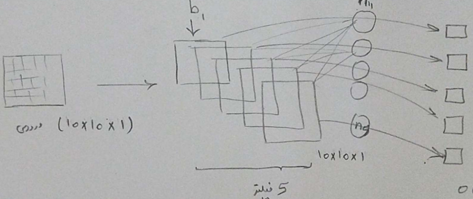
Fully connected

$g(x_1 w_{11} + x_2 w_{21} + \dots + x_{100} w_{100 \times 1} + b_1) \rightarrow$ output of n_1

$100 + 1 = 101$ تعداد پارامترها به ازای هر خروجی

$101 \times 5 = 505$ تعداد کل پارامترها

Method 2:



$S(x_{11} w'_{11} + x_{12} w'_{12} + \dots x_{1 \times 10} w'_{1 \times 10} \dots x_{10 \times 10} \times w'_{10 \times 10} + b_1) \rightarrow$ output of n_1

$100 + 1 = 101$ تعداد پارامترها به ازای هر خروجی

$101 \times 5 = 505$ تعداد کل پارامترها

با شرایط ذکر شده در صورت مسأله تعداد وزن هایی که می بایست آموزش دیده شوند در هر دو متد یکسان می باشد لذا از این جهت هر دو متد یکسان می باشند تنها تفاوت در activation fun. ها می تواند باشد و همچنین شکل ورودی

با توجه به اینکه softmax عملکرد بهتری نسبت به تابع threshold ذکر دارد به نظر می رسد علی رغم پارامترهای یکسان و معادلات یکسان در آموزش هر دو متد، همچنان متد 2 CNN بهتر عمل نماید (ورودی تصویر + softmax func)

تمرین 5)

$$\mathcal{L}_{CE}(\hat{y}, y) = -\sum_{i=1}^{n_y} y_i \log \hat{y}_i$$

$n_y = (\text{dog}, \text{Cat}, \text{iguana}, \text{mouse})$

قسمت اول a)

a) $\begin{cases} \hat{y} = (0.25, 0.25, 0.3, 0.2)^T \\ y = (0, 0, 1, 0)^T \end{cases} \Rightarrow \boxed{\mathcal{L}_{CE} = -\log 0.3}$

$$\hat{y} = (0, 0, 0.4, 0.6) \Longrightarrow \mathcal{L}_{CE} = -(0 \times \log 0 + 0 \log 0 + 1 \log 0.4 + 0 \log 0)$$

$$\boxed{= -\log 0.4}$$

قسمت دوم b)

$$-\log 0.4 < -\log 0.3$$

قسمت سوم c)

خطا در قسمت دوم را کمتر محاسبه کرده در صورتیکه در قسمت دوم misprediction داشتیم و این بدین دلیل است که میزان خطا را فقط در مورد خروجی مورد انتظار محاسبه می کند. زیرا مقدار y در بقیه خروجی ها صفر می باشد

$y \log \hat{y}$ ← $(0, 0, 1, 0)$

مقدار y برای سه خروجی معرف dog, Cat, mouse صفر می باشد بنابراین در قسمت دوم شبکه قادر به تصحیح خطا در تشخیص نادرست mouse نخواهد بود.

برای در نظر گرفتن خطا در همه خروجی ها لازم است به فرمول مذکور عبارت $(1-y)\log(1-\hat{y})$ اضافه گردد این عبارت خطا را در خروجی های دیگر وقتی $y=0$ است محاسبه می کند

$$-\sum_{i=1}^{n} y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)$$

قسمت چهارم d)

استفاده از دقت یا accuracy به جای تابع cross-entropy loss ایده خوبی نمی باشد زیرا در مرحله backpropagation نیاز به تابعی داریم که قابل مشتق گیری باشد تا بتوان با آن W ها را update نمود علاوه تابع accuracy طبق تعریف شده در مسأله برای هر ورودی یا Example به تنهایی قابل محاسبه نمی باشد و بعد از try کردن کل دیتاست قابل محاسبه می باشد.

(e) تابع Softmax یک تابع احتمال به صورت زیری باشد

$$S(y_i) = \frac{e^{y_i}}{e^{y_1} + e^{y_2} + \cdots + e^{y_i} + \cdots + e^{y_K}}$$

وقتی که تعداد خروجی ها K باشد

این تابع هیچ وقت ۰ یا ۱ نمی‌گردد زیرا طبق بیان خلف

$$\text{if } \begin{cases} (S(y_i) = 1 \Rightarrow e^{y_j} = 0 \quad \forall j \neq i & \times \\ \text{if } (S(y_i) = 0 \Rightarrow e^{y_i} = 0 & \times \end{cases}$$

اما تابع نمایی هیچ وقت صفر نمی‌گردد

بنابراین $\hat{y}$ به وارد تابع loss func می‌گردد هیچ وقت ۱ نمی‌گردد پس $y_i \log \hat{y}_i$ وقتی $y_i = 1$ محاسبه شده $\log \hat{y}_i$ خلاف صفر است (چون $\hat{y}_i \neq 1$) و هیچ خطا می‌تواند به صفر نزدیک شود اما هیچ‌گاه صفر نمی‌گردد.

(a) تعداد پارامترهای شبکه A برای tune شدن کمتر از پارامترهای شبکه B می باشد

(b) شبکه B هیچ ارجحیتی نسبت به شبکه A ندارد با توجه به اینکه activation function در هر دو شبکه خطی می باشد عمق بیشتر شبکه در آموزش توابع غیر خطی تأثیری ندارد و بی فایده است

(i) $\delta_1 = \frac{\partial E(w)}{\partial z_1^2} = \frac{\partial E(w)}{\partial a_1^2} \times \frac{\partial a_1^2}{\partial z_1^2} = \omega_1^3 \times a_1^2 (1 - a_1^2)$

(ii) $\delta_2 = \frac{\partial E(w)}{\partial z^2}$

$\frac{\partial E(w)}{\partial z_1^2} = \omega_1^3 \times a_1^2 (1 - a_1^2)$

$\frac{\partial E(w)}{\partial z_2^2} = \omega_2^3 \times a_2^2 (1 - a_2^2)$

(iii) $\delta_3 = \frac{\partial E(w)}{\partial z^1} =$

$\frac{\partial E(w)}{\partial z_1^1} = \frac{\partial E(w)}{\partial a_1^2} \times \frac{\partial a_1^2}{\partial z_1^2} \times \frac{\partial z_1^2}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial z_1^1} + \frac{\partial E(w)}{\partial a_2^2} \times \frac{\partial a_2^2}{\partial z_2^2} \times \frac{\partial z_2^2}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial z_1^1}$

$= \omega_1^3 \times a_1^2 (1 - a_1^2) \times \omega_{11}^2 \times a_1^1 (1 - a_1^1) + \omega_2^3 \times a_2^2 (1 - a_2^2) \times \omega_{21}^2 \times a_1^1 (1 - a_1^1)$

$\frac{\partial E(w)}{\partial z_2^1} = \frac{\partial E(w)}{\partial a_2^2} \times \frac{\partial a_2^2}{\partial z_2^2} \times \frac{\partial z_2^2}{\partial a_2^1} \times \frac{\partial a_2^1}{\partial z_2^1} + \frac{\partial E(w)}{\partial a_1^2} \times \frac{\partial a_1^2}{\partial z_1^2} \times \frac{\partial z_1^2}{\partial a_2^1} \times \frac{\partial a_2^1}{\partial z_2^1}$

$\omega_2^3 \times a_2^2 (1 - a_2^2) \times \omega_{22}^2 \times a_2^1 (1 - a_2^1) + \omega_1^3 * a_1^2 (1 - a_1^2) \times \omega_{12}^2 \times a_2^1 (1 - a_2^1)$

(iv) $\delta_4 = \frac{\partial E(w)}{\partial \omega_{11}^1} = \frac{\partial E(w)}{\partial a_1^2} \times \frac{\partial a_1^2}{\partial z_1^2} \times \frac{\partial z_1^2}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial z_1^1} \times \frac{\partial z_1^1}{\partial \omega_{11}^1} +$

$\frac{\partial E(w)}{\partial a_2^2} \times \frac{\partial a_2^2}{\partial z_2^2} \times \frac{\partial z_2^2}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial z_1^1} \times \frac{\partial z_1^1}{\partial \omega_{11}^1} =$

$\omega_1^3 \times a_1^2 (1 - a_1^2) \times \omega_{11}^2 + a_1^1 (1 - a_1^1) + a_1 +$

$\omega_2^3 \times a_2^2 (1 - a_2^2) \times \omega_{21}^2 \times a_1^1 (1 - a_1^1) \times a_1$