حمیدرضا امیرزاده
۴۰۱۲۰۶۹۹۹

تمارین سری سوم درس Deep Learning

۱) الف) با توجه به نمودار داده شده می‌توان دید که با شروع از هر مقداری برای h_0، به صورت آمدن h_1، سپس به همین صورت به دست آمدن h_2، h_3، ...، h_∞، واضح است که h_∞ به مقدار ۰.۸ میل می‌کند. به عبارت دیگر اگر h_0 کوچکتر از ۰.۸ باشد آنگاه با دقت به نمودار داده شده می‌توان دید که h_∞ به ۰.۸ میل می‌کند. همین روند برای وقتی که h_0 بزرگتر از ۰.۸ باشد نیز تکرار می‌شود.

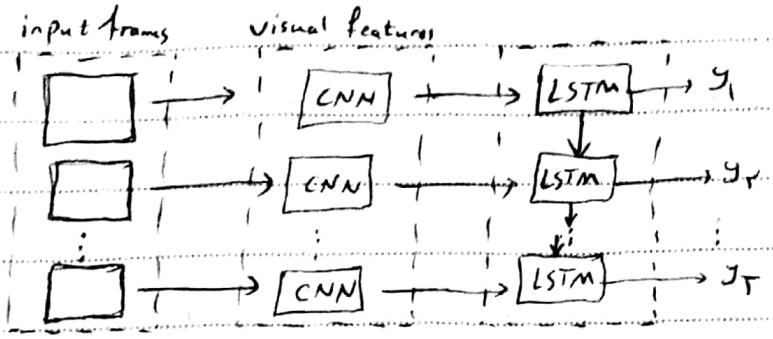
ب) از آنجایی که با انتخاب هر مقداری برای h_0، در نهایت h_t به سمت ۰.۸ میل می‌کند و داریم $\frac{\partial h(t+1)}{\partial h(t)} = 3h(t+1)(1-h(t+1))$ بنابراین می‌توان گفت که به ازای هیچ مقداری از h_0، در این شبکه مشکل انفجار یا محو شدن گرادیان رخ نمی‌دهد.

ج) همانگونه که توضیح داده شد مشخص می‌گردد که نقطه تعادل برابر با $h_t = 0.8$ می‌باشد. حال برای اینکه مشخص کنیم این نقطه تعادل، source است یا sink باید مشتق h_{t+1} نسبت به h_t را در نقطه ۰.۸ حساب کنیم. بعد از آن مشخص می‌گردد که مشتق در این نقطه منفی است، بنابراین نقطه ۰.۸ یک نقطه تعادل sink می‌باشد.

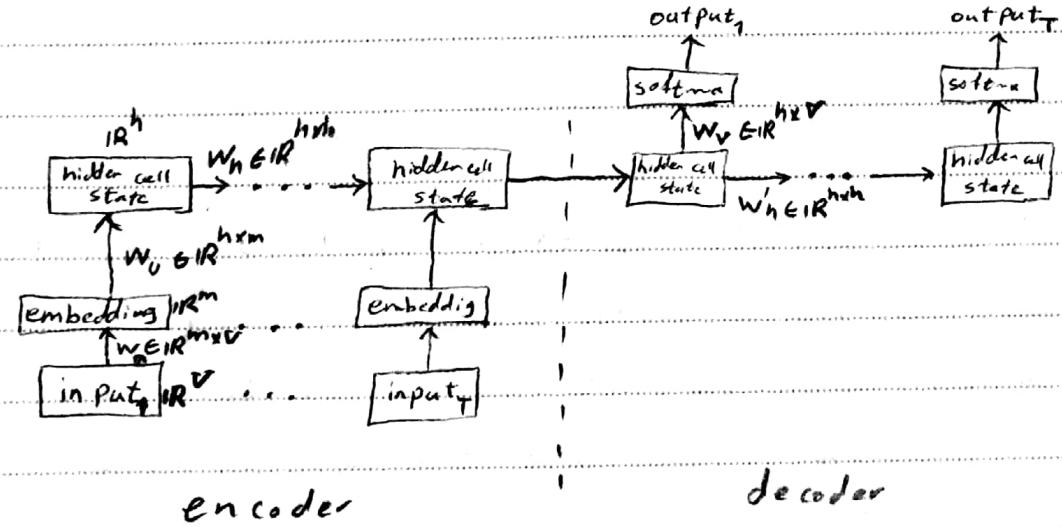
د) با توجه به رابطه $h_{t+1} = Relu(3h_t - 1)$ مشخص است که اگر h_0 بزرگتر از ۰.۵ باشد آنگاه با جایگذاری های متعدد h_t به سمت ∞ میل می‌کند و مشکل explode وجود دارد. حال اگر h_0 کوچکتر از ۰.۵ باشد آنگاه با جایگذاری های متعدد h_t بعد از چند مرحله برابر صفر می‌شود و مشکل Vanish پدید می‌آید. به ازای h_0 برابر ۰.۵، مقدار h_t همواره برابر ۰.۵ باقی می‌ماند.

$$h_t = Relu\left(3^t h_0 - \frac{3^t - 1}{2}\right)$$

(۲) الف) وان انجام تک دسته‌بندی ویدئو می‌توان هر فریم از ویدئو را به یک شبکه CNN داد و سپس بردار بازنمایی حاصل را به یک LSTM داد.



ب) برای انجام این تسک، یک شبکه encoder decoder به صورت زیر معرفی می‌کنیم. رشته ورودی را به encoder داده (البته ابتدا با استفاده از یک embedding رشته ورودی را به یک بردار تبدیل می‌کنیم). سپس cell state نهایی آفر این encoder را به عنوان ورودی اولیه به شبکه decoder می‌دهیم. وظیفه این شبکه بازسازی تقریبی از رشته‌ای که به آن داده شده است. لازم به ذکر است که در شبکه‌های encoder و decoder گفته شده از طبقه‌های RNN استفاده می‌کنیم.



i) برای محاسبه تعداد کل محاسبات داریم:

تعداد محاسبات به دست آمده $= m(V + V-1) = m(2V-1)$

embedding

ضرب، جمع

تعداد محاسبات به دست آمده

$h_t = \sigma(W_U X + W_h h_{t-1} + bias)$

$\mathbb{R}^{h \times m}$ $\mathbb{R}^{m}$ $\mathbb{R}^{h \times h}$ $\mathbb{R}^{h}$ $\mathbb{R}^{h}$

hidden state

$h(2m-1)$ $h(2h-1)$ h

sigmoid تابع σ = محاسبات $3h$

$\Rightarrow 2mh + 2h^2 + 2h$

بنابراین تعداد محاسبات دارد encoder برای آن با:

$[2mh + 2h^2 + 3h + 2mV - m]T$

برای محاسبات decoder، تعداد محاسبات به دست آمده hidden state به encoder برای با $2h^2 + 2h$

با بدون داشتن ورودی

هستند. اما برای انجام محاسبات تولید خروجی داریم:

$\mathbb{R}^{h \times V}$ $\mathbb{R}^{h}$ $\mathbb{R}^{h}$

$\text{softmax}: y_i = \text{softmax}(W_V h_i + bias)$

$V(2h-1)$ V

softmax تابع = $3V$

$\Rightarrow 2Vh + 3V$

بنابراین تعداد محاسبات دارد decoder برای آن با:

$[2h^2 + 4h + 2Vh + 3V]T$

بنابراین تعداد کل محاسبات برای آن با:

$[4h^2 + Vh + 2mh + 2mV + 2Vh - m + 3V]T$

ii) برای محاسبه تعداد پارامترها داریم:

W_m W_U W_h بایاس

تعداد پارامترهای encoder: $mxV + hxm + hxh + h$

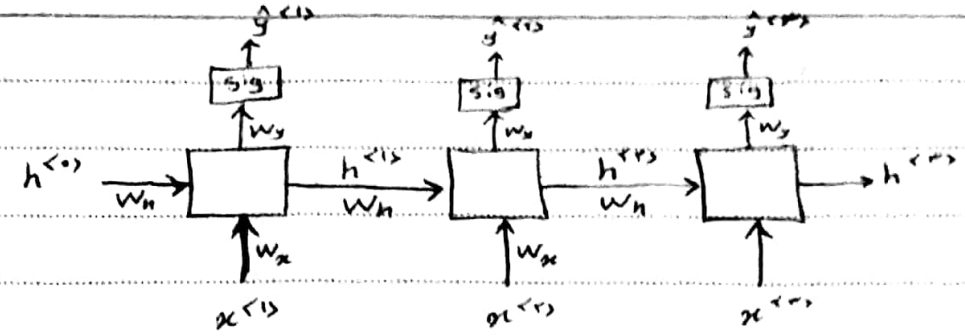
تعداد پارامترهای decoder: $hxh + hxV + h + V$

W_h' W_V بایاس hidden state بایاس خروجی

تعداد کل پارامترها

$= mV + mh + 2h^2 + hV + 2h + V$

(۳) الف)



$$h^{<t>} = W_h h^{<t-1>} + W_x x^{<t>} \qquad \hat{y}^{<t>} = \sigma(W_y h^{<t>})$$

$$E^{<t>} = (y^{<t>} - \hat{y}^{<t>})^2$$

$$\left.\begin{aligned} E^{<3>} &= (y^{<3>} - \hat{y}^{<3>})^2 \\ \hat{y}^{<3>} &= \sigma(W_y h^{<3>}) \end{aligned}\right\} \Rightarrow E^{<3>} = \left(y^{<3>} - \sigma(W_y h^{<3>})\right)^2$$

از آنجایی که W_y در تابع $h^{<3>}$ نقشی ندارد داریم:

$$\frac{\partial E^{<3>}}{\partial W_y} = -2\left(y^{<3>} - \sigma(W_y h^{<3>})\right) h^{<3>}\, \sigma(W_y h^{<3>})\left(1 - \sigma(W_y h^{<3>})\right)$$

$$\frac{\partial E^{<3>}}{\partial W_y} = -2\left(y^{<3>} - \hat{y}^{<3>}\right) \hat{y}^{<3>}\left(1 - \hat{y}^{<3>}\right) h^{<3>}$$

$$\frac{\partial E^{<3>}}{\partial W_h} = \frac{\partial E^{<3>}}{\partial \hat{y}^{<3>}} \frac{\partial \hat{y}^{<3>}}{\partial h^{<3>}} \left(\sum_{i=1}^{3} \frac{\partial h^{<3>}}{\partial h^{<i>}} \frac{\partial h^{<i>}}{\partial W_h} \right)$$

$$\frac{\partial h^{<3>}}{\partial W_h} = \sum_{i=1}^{3} \frac{\partial h^{<3>}}{\partial h^{<i>}} \frac{\partial h^{<i>}}{\partial W_h}$$

$$h^{<3>} = W_h h^{<2>} + W_x x^{<3>}$$

$$= W_h\left(W_h h^{<1>} + W_x x^{<2>}\right) + W_x x^{<3>}$$

$$= W_h^2 h^{<1>} + W_h W_x x^{<2>} + W_x x^{<3>}$$

$$= W_h^2\left(W_h h^{<0>} + W_x x^{<1>}\right) + W_h W_x x^{<2>} + W_x x^{<3>}$$

$$= W_h^3 h^{<0>} + W_h^2 W_x x^{<1>} + W_h W_x x^{<2>} + W_x x^{<3>}$$

$$\frac{\partial h^{(3)}}{\partial h^{(2)}} = W_h^2 \quad , \quad \frac{\partial h^{(2)}}{\partial h^{(1)}} = W_h \quad , \quad \frac{\partial h^{(1)}}{\partial h^{(0)}} = 1$$

$$\frac{\partial h^{(1)}}{\partial W_h} = h^{(0)} \quad , \quad \frac{\partial h^{(2)}}{\partial W_h} = 2W_h h^{(0)} + W_x x^{(1)} \quad , \quad \frac{\partial h^{(3)}}{\partial W_h} = 3W_h^2 h^{(0)} + 2W_h W_x x^{(1)} + W_x x^{(2)}$$

$$\Rightarrow \sum_{i=1}^{3} \frac{\partial h^{(3)}}{\partial h^{(i)}} \frac{\partial h^{(i)}}{\partial W_h} = W_h^2 h^{(0)} + 2W_h^2 h^{(0)} + W_h x^{(1)} + 2W_h^2 h^{(0)} + 2W_h W_x x^{(1)} + W_x x^{(2)}$$

$$= 6W_h^2 h^{(0)} + W_h (2W_x x^{(1)} + x^{(1)}) + W_x x^{(2)}$$

$$\frac{\partial E^{(3)}}{\partial \hat{y}^{(3)}} = 2(\hat{y}^{(3)} - y^{(3)})$$

$$\frac{\partial \hat{y}^{(3)}}{\partial h^{(3)}} = W_y \hat{y}^{(3)} (1 - \hat{y}^{(3)})$$

$$\Rightarrow \frac{\partial E^{(3)}}{\partial W_h} = 2W_y \hat{y}^{(3)} (\hat{y}^{(3)} - y^{(3)})(1 - \hat{y}^{(3)})(6W_h^2 h^{(0)} + W_h (2W_x x^{(1)} + x^{(1)}) + W_h x^{(2)})$$

$$\frac{\partial E^{(3)}}{\partial W_x} = \frac{\partial E^{(3)}}{\partial \hat{y}^{(3)}} \frac{\partial \hat{y}^{(3)}}{\partial h^{(3)}} \left(\sum_{i=1}^{3} \frac{\partial h^{(3)}}{\partial h^{(i)}} \frac{\partial h^{(i)}}{\partial W_x} \right)$$

$$\frac{\partial h^{(1)}}{\partial W_x} = x^{(1)} \quad , \quad \frac{\partial h^{(2)}}{\partial W_x} = W_h x^{(1)} + x^{(2)} \quad , \quad \frac{\partial h^{(3)}}{\partial W_x} = W_h^2 x^{(1)} + W_h x^{(2)} + x^{(3)}$$

$$\Rightarrow \frac{\partial E^{(3)}}{\partial W_x} = 2W_y \hat{y}^{(3)} (\hat{y}^{(3)} - y^{(3)})(1 - \hat{y}^{(3)})(3W_h^2 x^{(1)} + 2W_h x^{(2)} + x^{(3)})$$

ب) متاسفانه نمی‌دانم که در تابع گرادیان ما در مرحله BackProp برای برگشتن به یک لحظه قبلتر، ماتریس W_h در حاصل ضرب می‌شود بنابراین زمانی که گرادیان به لحظه اول می‌رسد در گرادیان حاصل یک حاصل ضرب می‌شود W_h^t داریم. در واقع: $h^{(t)} \propto Q^T \Lambda^t Q h^{(0)}$ که می‌توان نوشت: $h^{(t)} \propto W_h^t h^{(0)}$

حال اگر بزرگترین Singular value ماتریس W_h بزرگتر از یک باشد ← explode و اگر کوچکتر از یک باشد ← Vanish

$h^{<t>} = W^T h^{<t-1>}$ $\quad W = \begin{bmatrix} ۰,۵۸ & ۰,۲۴ \\ ۰,۲۴ & ۰,۷۲ \end{bmatrix}$ $\quad h^{<۰>} = I_۲$ (۴)

(الف

$det(W - \lambda I) = ۰$

$(۰,۵۸ - \lambda)(۰,۷۲ - \lambda) - ۰,۲۴^۲ = ۰ \Rightarrow \lambda_۱ = ۰,۴ \;,\; \lambda_۲ = ۰,۹$

$(W - \lambda_۱ I) X_۱ = ۰ \Rightarrow \begin{bmatrix} ۰,۱۸ & ۰,۲۴ \\ ۰,۲۴ & ۰,۳۲ \end{bmatrix} x_۱ = ۰$

$\Rightarrow x_۱ = \begin{bmatrix} -۰,۸ \\ ۰,۶ \end{bmatrix}$

$(W - \lambda_۲ I) X_۲ = ۰ \Rightarrow \begin{bmatrix} -۰,۴۲ & ۰,۲۴ \\ ۰,۲۴ & -۰,۱۸ \end{bmatrix} x_۲ = ۰$

$\Rightarrow x_۲ = \begin{bmatrix} ۰,۶ \\ ۰,۸ \end{bmatrix}$

$W = Q \Lambda Q^{-1}$ $\quad Q = \begin{bmatrix} -۰,۸ & ۰,۶ \\ ۰,۶ & ۰,۸ \end{bmatrix}$

$\Lambda = \begin{bmatrix} ۰,۴ & ۰ \\ ۰ & ۰,۹ \end{bmatrix}$

$\Rightarrow Q^{-1} = Q^T \Rightarrow W = Q \Lambda Q^T \Rightarrow W^T = Q^T \Lambda Q$

$h^{<t>} = W^T h^{<t-1>} \Rightarrow h^{<t>} = (Q^T \Lambda Q) h^{<t-1>}$

$\Rightarrow h^{<۳۰>} = Q^T \Lambda^{۳۰} Q = \begin{bmatrix} -۰,۸ & ۰,۶ \\ ۰,۶ & ۰,۸ \end{bmatrix} \begin{bmatrix} ۰,۴^{۳۰} & ۰ \\ ۰ & ۰,۹^{۳۰} \end{bmatrix} \begin{bmatrix} -۰,۸ & ۰,۶ \\ ۰,۶ & ۰,۸ \end{bmatrix}$

موضوع .....

تاریخ .....

ب) همانگونه که مشاهده می‌شود داریم:

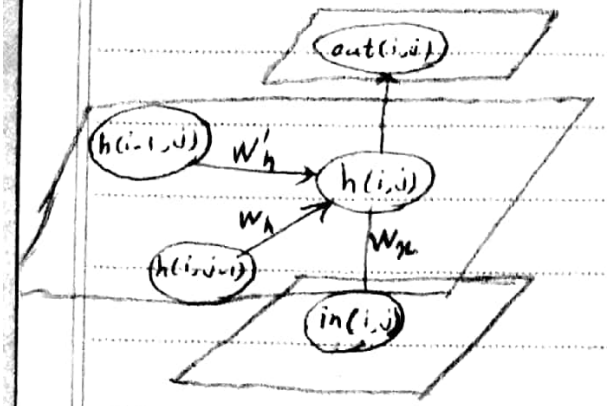
$$h^{<t>} = Q^T \Lambda^t Q$$

از آنجایی که داشتیم $\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 0.9 \end{bmatrix}$ بنابراین ماتریس Λ^t با بزرگ شدن t ...

ماتریس‌ها صفر سیل می‌کنند. بنابراین می‌توان گفت این نوع از شبکه بازگشتی با افزایش تعداد تمالی ورودی‌ها نمی‌تواند خروجی مناسبی تولید کند. همچنین به‌علتی مشابه در فاز Backward باعث Vanish شدن گرادیان و عدم آپدیت وزن‌ها می‌شود.

(۴) در شبکه های RNN تک بعدی ، توانایی پردازش داده های فقط یک بعدی مثل توالی از کلمات یا توالی از اعداد وجود دارد و امکان پردازش داده های چند بعدی مثل تصویر و ویدیو وجود ندارد. شبکه های MDRNN قابلیت پردازش داده های چند بعدی را دارند. ایده ی آن ها به این صورت است که یک اتصال بازگشتی واحد که در RNN ها وجود دارد را با n اتصال بازگشتی جایگزین کند که n برای تعداد بُعد ورودی است. در این نوع شبکه ها در زمان forword علاوه بر ورودی هر نقطه ، خروجی های لایه های پنهان هر یک از بعد های دیگر نیز داخل می شوند

به عنوان مثال اگر داده ورودی ۲ بعدی باشد ($n=2$) نحوه بدست آوردن مقدار hidden state در هر نقطه به صورت زیر است:

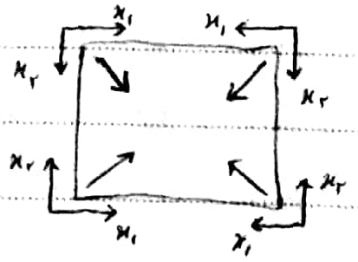


$$h(i,j) = W_x x(i,j) + \underbrace{W_h' h(i-1,j)}_{\text{if } i>0} + \underbrace{W_h h(i,j-1)}_{\text{if } j>0}$$

الف) اگر بخواهیم مفهوم MDRNN را با مفهوم BLSTM ترکیب کنیم به مفهوم Multi Dimensional Bidirectional LSTM می رسیم.

در BLSTM به این صورت است که در مرحله forword داده های توالی دیده شده از اول و هم از آخر به اول پردازش می شوند. برای ساختن خروجی در هر نقطه از دو hidden state که یکی مربوط به جهت ← و یکی مربوط به جهت → می باشند استفاده می کنیم.

حال ساختار Multi Dimensional LSTM به این صورت است که در هر LSTM Unit به جای یک گیت forget به تعداد ابعاد ورودی از این نوع گیت داریم. حال اگر بخواهیم ساختار 2D BLSTM را طبق شکل دهیم به این صورت است که در مرحله $2^n = 2^2 = 4$ hidden state دیده دارد که جهت های متناظر آن ها به صورت زیر است:



ب) برای تابع مربوط به forward در MORNN، ابتدا روی تمامی ابعاد یک حلقه for می‌زنیم تا بتوانیم روی تمامی داده‌ها و در تمامی ابعاد آن‌ها را داشته باشیم. سپس هر یک از این داده‌ها را با ماتریس وزن W_x ضرب می‌کنیم. سپس برای تمامی ابعاد کناری این داده‌ها، ماتریس وزن W_h را در قبلی hidden state آن داده در بعد کناری ضرب می‌کنیم و این نتایج را با هم جمع می‌کنیم. برای همین مقدار hidden state های داده در بعد فعلی ما تعیین می‌شود. pseudo code مربوطه به صورت زیر است.

```
for i1 = 0 to I1 - 1
    for i2 = 0 to I2 - 1
        ...
            for in = 0 to In - 1
                b ← Wx x(i1, i2, ..., in)
                for j = 1 to n
                    if ij > 0
                        b ← b + W_h^j h(i1, ..., ij - 1, ..., in)
                h(i1, i2, ..., in) ← g(b)
                ↓
            تابع فعال ساز
```

* از آنجایی که n بعد داریم و برای جمع hidden state در هر لحظه نیاز به پیدا کردن تمامی hidden state های قبلی برحسب تمامی n بعد داریم. بنابراین وزن‌هایی که وجود دارند عبارت است از یک وزن W_x که عضو $\mathbb{R}^{in \times h}$ است، n وزن W_h نیاز که هرکدام عضو $\mathbb{R}^{h \times h}$ هستند و یک وزن W_y که عضو $\mathbb{R}^{h \times out}$ است. برحسب مثال یک بایاس $\mathbb{R}^h$ و n بایاس $\mathbb{R}^h$ و یک بایاس $\mathbb{R}^{out}$ داریم. بنابراین تعداد کل پارامترها عبارت است از:

$$in \times h + n(h \times h) + h \times out + h + n \times h + out$$

حال اگر ما بتوانیم ماتریس‌هایی داشته باشیم که بتوان عملیات مربوط به ابعاد مختلف را به یک ماتریس نمایش داده کرد تا این عملیات برحسب سرعت و حجم انجام شوند. از طرفی با توان عملیات مربوط به پیدا کردن و ذخیره در لحظات مختلف را نیز سرعت انجام داد. نقطه عملیات مربوط به hidden state را می‌توان سرعت انجام داد زیرا به لحظه قبل بستگی دارد. بنابراین سرعت زمان اجرای کلی متناسب است با حاصل ضرب دو بعدی.