Assignment NO.9 Solutions

Deep learning | spring 1401 | Dr.Mohammadi

Teaching Assistant:

Hadise Mahmoodi

Student name : **Amin Fathi**

Student id : **400722102**

Problem1

نگارندگان مقاله نام روش نوآورانه خود را RNN Encoder-Decoder نامگذاری کرده اند که در واقع از دو شبکه بازگشتی تشکیل شده است ؛ دو شبکه ای که به مصابه !!! encoder و decoder عمل می کنند ، encoder در واقع دنباله ای به طول متغییر (مثلا یک جمله) را به یک بردار به طول ثابت نگاشت می کند و decoder هم بردار خروجی را به دنباله ی هدف (مثلا ترجمه جمله گفته شده) با طول متغییر تبدیل می کند ، همچنین در معماری ذکر شده از یک مجموعه نورون ها لایه پنهان نسبتا پیچیده برای سهولت در محاسبات و تقویت حافظه هم استفاده می شود ، نگارندان مقاله ایده ذکر شده را بر روی پروژه ترجمه متن از انگلیسی به فارسی پیاده سازی کرده اند که به نتایج قابل توجهی هم دست یافتند .
از منظر احتمالی، این مدل جدید یک روش کلی برای یادگیری توزیع شرطی روی یک دنباله با طول متغیر است که البته فضای خروجی و ورودی می تواند ابعاد متفاوتی داشته باشند .

بخش encoder در واقع یک RNN است که هر نشانه (سیمبل) در ورودی را میخواند و سپس نورون لایه میانی مربوطه را آپدیت میکند با فرمول زیر که فرمول عام RNN است :

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right),$$

این بخش از مدل کارش را ادامه می دهد تا زمانی که نشانه (سیمبل) مربوط به پایان متن را مشاهده کند اخرین نورون لایه پنهان را آپدیت میکند که در واقع خلاصه ورودی هاست و آن را با C نشان می دهد .

قسمت decoder هم یک RNN است که آموزش میبیند تا رشته خروجی را با استفاده از نورون های لایه پنهان decoder تولید کند و تفاوتش با RNN معمولی در این است که در واقع h(t) علاوه بر h(t-1) به خروجی قبلی(حرف یا کلمه قبلی) y(t-1) و همچنین خلاصه رشته ورودی یعنی c هم بستگی دارد :

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c}\right),$$

شمای آنچه گفته شد را در تصویر زیر می توانید مشاهده کنید :

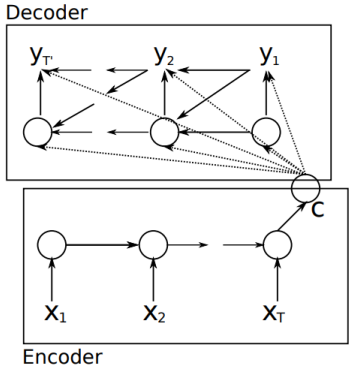


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

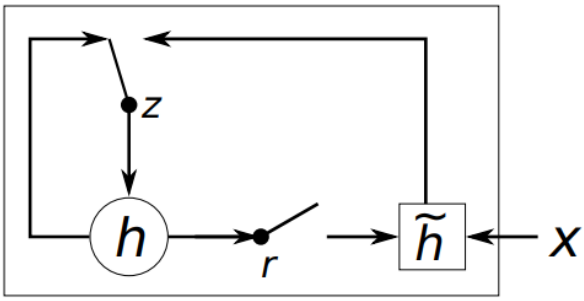
لازم به ذکر است که encoder و decoder به طور مشترک train می شوند تا پارامتر های مناسب برای بیشنیه سازی توزیع شرطی حاصل شود :

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g\left(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c}\right)$$

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{y}_n \mid \mathbf{x}_n),$$

پس از آموزش دیدن مدل ، مدل را می توان به دو شکل به کار برد ؛ یکی اینکه یک رشته خروجی را با توجه به رشته ورودی و دریافتی تحویل دهد (مثل ترجمه از یک زبان به یک زبان دیگر) روش دیگر اینکه از این مدل برای امتیاز دادن (score) به جفت رشته های ورودی و خروجی داده شده .

در مورد لایه پنهان به کار رفته در این مدل هم می توان ابتدا به شمای گرافیکی آن که در زیر آمده است اشاره کرد :



حال رابطه زیر را در نظر بگیرید

$$r_j = \sigma\left(\left[\mathbf{W}_r\mathbf{x}\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1 \rangle}\right]_j\right),$$

که در آن از تابع سیگموید استفاده شده است و ورودی تابع سیگموید

در واقع زمانی که reset gate نزدیک به صفر است ، نود لایه پنهان مجبور می شودکه لایه های پنهان قبلی را فراموش کند و به آن ها بی توجه باشد و فقط با ورودی فعلی مقدار خود را تعیین کند که این عمل باعث می شود شبکه بتواند در صورتی که اطلاعات قبلی که در آینده به درد نخواهند خورد را استفاده نکنیم و سربار حافظه و محاسبات کمتری هم داشته باشیم

فرمول مربوط به reset gat به شکل زیر است که در آن سیگما معرف تابع سیگموید است و Wr , Ur هم وزن هایی است که در مرحله قبل آموزش داده ایم .

در مورد update gate هم می توان گفت که مشخص کننده این است که چه مقدار اطلاعات از لایه های قبلی قرار است در محاسبه مقدار نود لایه پنهان فعلی استفاده شود که در واقع شبیه سلول های حافظه در LSTM است

روابط مربوط به این قسمت را در زیر مشاهده می کنید :

$$z_j = \sigma\left(\left[\mathbf{W}_z\mathbf{x}\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right).$$

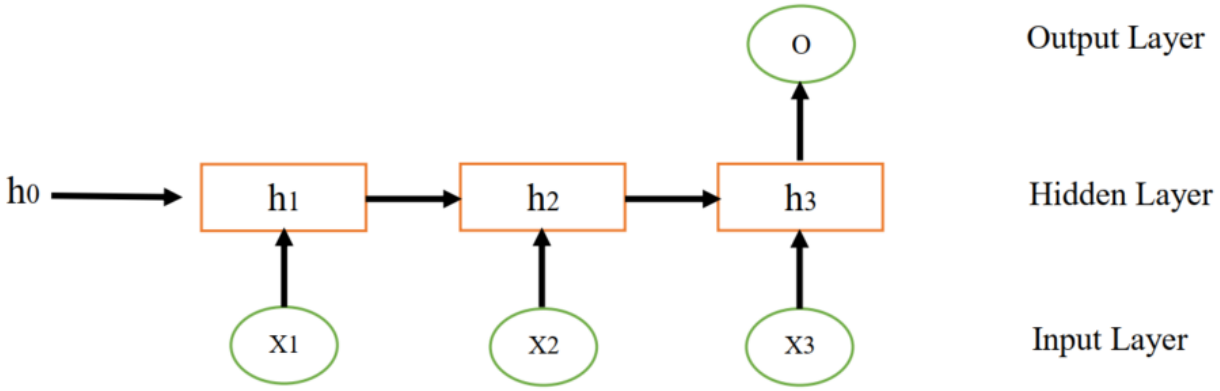
$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1 - z_j)\tilde{h}_j^{\langle t\rangle},$$

$$\tilde{h}_j^{\langle t\rangle} = \phi\left(\left[\mathbf{W}\mathbf{x}\right]_j + \left[\mathbf{U}\left(\mathbf{r}\odot\mathbf{h}_{\langle t-1\rangle}\right)\right]_j\right)$$

لازم به ذکر است که هر نود دارای گیت های reset , update جداگانه مربوط به خود است و مشخصا نود هایی که به دنبال یادگیری اطلاعات short term هستند ، reset gate های فعال تری دارند و نود هایی که به دنبال به کار بردن اطلاعات گذشته و یا همان long term dependencies هستند update gate های فعالتری خواهند داشت .

Problem2

الف) هدف از این تمرین آشنایی باBackpropagation در شبکه های بازگشتی است. معماری داده شده یک شبکه ساده بازگشتی را نشان می دهد که لایه های ورودی، پنهان و خروجی در آن مشخص شده است. تابع ضرر را MSE در نظر بگیرید. تمام مراحل به روزرسانی را برای دو وزن Wxh وWhh به صورت دستی بنویسید. (وزن Wxh و وزن Whh به ترتیب در لایه ورودی و لایه پنهان مشترک هستند. از تابع فعالسازی صرف نظر کنید)



پاسخ :

از خطای MSE استفاده شده است فلذا تابع خطا به شکل زیر خواهد بود ، لازم به ذکر است d همان desire یا مقدار مطلوب مان جهت مقایسه با خروجی o می باشد .

$$E = (d - o)^2$$

در این صورت مقدار گرادیان خطا نسبت به وزن Whh برابر خواهد بود با :

$$\frac{\partial E}{\partial w_{hh}} = \left(\frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_{hh}}\right) + \left(\frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{h_2} \cdot \frac{\partial h_2}{\partial w_{hh}}\right)$$

$$+ \left(\frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_{hh}}\right)$$

سپس با استفاده از این گرادیان مقادیر وزن ها را اپدیت می کنیم .

$$W_{hh_{new}} = W_{hh_{old}} - \eta \frac{\partial E}{\partial w_{hh}}$$

مقدار گرادیان نسبت به وزن Wxh هم برابر است با :

$$\frac{\partial E}{\partial w_{xh}} = \left(\frac{\partial E}{\partial O} \cdot \frac{\partial O}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_{xh}}\right) + \left(\frac{\partial E}{\partial O} \cdot \frac{\partial O}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_{xh}}\right) + \left(\frac{\partial E}{\partial O} \cdot \frac{\partial O}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_{xh}}\right)$$

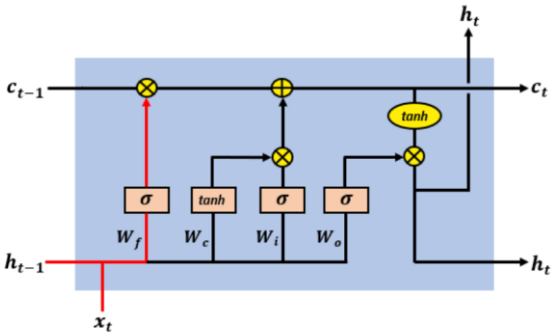
به طریقه مشابه وزن های Wxh را آپدیت می کنیم .

$$W_{xh_{new}} = W_{xh_{old}} - \eta \frac{\partial E}{\partial w_{xh}}$$

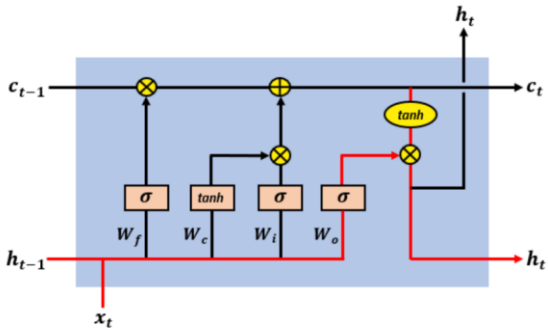
ب) مشکل Vanishing gradient در شبکه های بازگشتی را توضیح دهید و روش های حل آنرا نام ببرید

در شبکه های بازگشتی دو فاکتور داریم که بر روی بزرگی گرادیان ها اثر دارند ، یکی وزن ها و دیگری تابع فعالسازی ها (یا به طور مشخص تر مشتقات مربوط به توابع فعال سازی) همانطور که می دانیم که در RNN برای پیش بینی یک خروجی غالبا از یک تابع فعال سازی سیگموئید استفاده می کنیم تا بتوانیم خروجی احتمال را برای یک کلاس خاص بدست آوریم. و صد البته مساله ای که در RNN با آن مواجهیم زنجیره طولانی مشتقات بنابه ساختار معماری خود شبکه RNN است. حال شرایطی را در نظر بگیرید که مشتق سیگموید را می‌گیریم و مشتق سیگموئید همیشه زیر ۰,۲۵ باشد و از این رو وقتی مشتقات زیادی را طبق قانون زنجیره‌ای در یکدیگر ضرب می‌کنیم، با یک مقدار ناپدید می‌شویم به طوری که نمی‌توانیم از آنها برای محاسبه خطا استفاده کنیم. چرا که مقدار گرادیان کوچک است و به روز رسانی های پارامتر ناچیز می شود که به این معنی است که هیچ یادگیری واقعی انجام نمی‌شود. برای حل این مشکل از LSTM استفاده می شود یک شبکه LSTM دارای یک بردار ورودی [h(t-1)، x(t)]در مرحله زمانی t است. وضعیت سلول شبکه با c(t) نشان داده می شود. بردارهای خروجی عبور داده شده از شبکه بین مراحل زمانی متوالی t, t+1 با h(t) نشان داده می شوند. یک شبکه LSTM دارای سه گیت است که حالت های سلولی را به روز می کنند و کنترل می کنند، این ها دروازه فراموشی، گیت ورودی و گیت خروجی هستند.(input gate, output gate , forget gate) وجود این گیت ها که در شکل های زیر به صورت شماتیک توضیح داده شده اند ، باعث می

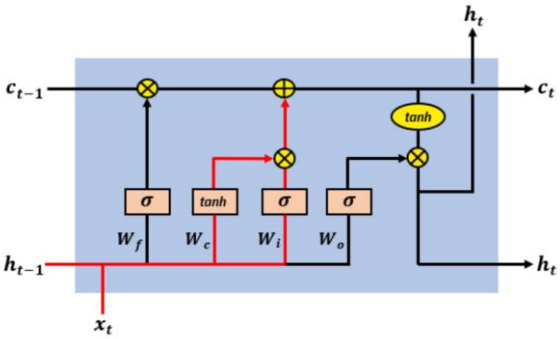
شود که در فرآیند backpropagation و در محاسبه گرادیان ها با جمع چند ترم رو به رو باشیم که این عمل در همان وهله اول به ما کمک می کند که با توجه به تعداد ترم ها از کم بودن مقدار گرادیان جلوگیری کنیم .



The LSTM forget gate update of the cell state



The LSTM output gate's action on the cell state



The LSTM input gate update of the cell state

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}[A_t + B_t + C_t + D_t]\right)\frac{\partial c_1}{\partial W}$$

$$A_t = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

مساله دیگری که می توان به آن اشاره کرد با توجه به فلسفه وجود forget gate می توان پارامتر های مدل را به نحوی تعیین کرد که مقدار گرادیان آنقدر ها کاهش نیابد ، به طور مثال فرض کنید این ترم در مقدار گرادیان forget gate برابر ۰ باشد :

$$\sum_{t=1}^{k} \frac{\partial E_t}{\partial W} \rightarrow 0$$

میتوان پارامتر های مربوط به k+1 برای forget fate را به نحوی محسابه کرد که مقدار گرادیان به ۰ میل نکند :

$$\frac{\partial E_{k+1}}{\partial W} \nrightarrow 0$$

دروازه فراموشی کنترل می کند که چه اطلاعاتی در حالت سلولی باید فراموش شود، با توجه به اطلاعات جدید وارد شده به شبکه.

منابع :

Let's Understand The Problems with Recurrent Neural Networks (analyticsvidhya.com)

Backpropagation Through Time for Recurrent Neural Network | Mustafa Murat ARAT (mmuratarat.github.io)

How LSTM networks solve the problem of vanishing gradients | by Nir Arbel | DataDrivenInvestor

Problem3

الف) دو مجموعه آموزشی و آزمایشی مشخص شده اند. باتوجه به جملات مجموعه آموزشی، احتمال Bigram جملات آزمایشی را محاسبه کنید. لطفا تمام مراحل محاسبات خود را یادداشت نمایید (تمام محاسبات به صورت دستی انجام شود)

Training corpus
<s> I like deep learning class </s>
<s> I am a student </s>
<s> student has different lessons </s>
<s> student likes learning different languages </s>

Test data
<s> I like learning different lessons </s>
<s> student likes deep learning class </s>

پاسخ :

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

P(<s> I like learning different lessons </s>) =

P(I | <s>) * P(like | I) * P(learning | like) * P(different | learning) * P(lessons | different) * P(</s>| lessons)

$= \frac{2}{4} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * 1 = \frac{1}{32}$

P(I|<s>) = $\frac{2}{4}$, P(like | I) = $\frac{1}{2}$, P(learning | like) = $\frac{1}{2}$, P(different | learning) = $\frac{1}{2}$

P(lessons | different) = $\frac{1}{2}$, P(</s>| lessons) = $\frac{1}{1}$

P(<s> student likes deep learning class </s>) =

P(student | <s>) * P(likes | student) * P(deep | likes) * P(learning | deep) * P(class | learning) * P(</s>| class)

$= \frac{2}{4} * \frac{1}{3} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * 1 = \frac{1}{24}$

P(student | <s>) $= \frac{2}{4}$, P(likes | student) $= \frac{1}{3}$ P(deep | likes)$= \frac{1}{2}$, P(different | learning) $= \frac{1}{2}$

P(lessons | different) $= \frac{1}{2}$, P(</s>| lessons) $= \frac{1}{1}$

ب) مدلهای مارکوف با مرتبه های ۰و۱و ۲را با یکدیگر مقایسه کنید. مزایا و معایب هرکدام را توضیح دهید

پاسخ :

مدل مارکوف مرتبه ۰ یا unigram مدلی است که در آن نمونه ها مستقل بوده و احتمال هر نمونه کاملا مستقل از بقیه نمونه ها به دست می آید .

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2)P(x_3)P(x_4)$$

مدل مارکوف مرتبه ۱ یا bigram مدلی که در آن هر نمونه از بین نمونه های گذشته فقط به نمونه قبلیش وابسته است .

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_3)$$

مدل مارکوف مرتبه ۲ یا tigram مدلی است که در آن هر نمونه به دو نمونه قبلی خود وابسته است .

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)P(x_4|x_2, x_3)$$

از آنجا که در unigram ها احتمال حضور هر کلمه در جمله فقط به همان کلمه وابسته است می توانیم با مشکلاتی رو به رو شویم از اصلی ترین مشکلات در این حالت این است که کلمات مسقلا در نظر گرفته می شوند و از لحاظ معنایی برای ما مشکل زا است . ولی خب در این حالت محاسبات آسانی داریم و احتمال شرطی کل مجموعه ۰ نمی شود .

به طور کلی در n-garam ها اگر مقدار n کم باشد در واقع داریم کاری میکنیم که هر جمله و پارت به تنهایی بررسی شود و نه در وابستگی با n بخش قبلی اش ، در این حالت داریم ارزش معنایی کلمات در مجاورت همدیگر را عملا نادیده میگیریم ، بالا

بردن n هم باعث می شود شروط سختگیرانه تری را برای ترکیب کلمات در نظر بگیریم که به طور مثال اگر مقدار n بسیار زیاد باشد ، nکلمه در کنار هم احتمال بسیار کمی دارند که با هم بیایند و این مقدار خروجی احتمال شرطی را کم میکند و حتی در این صورت احتمال کل جمله برابر ۰ خواهد بود . که البته برای حل این قضیه از :Laplace smoothing استفاده می کنیم .