Assignment NO.10 Solutions

Deep learning | spring 1401 | Dr.Mohammadi

Teacher Assistant:

Amir Mahdi nikokaran

Mohammad hossein khojaste

Student name : **Amin Fathi**

Student id : **400722102**

Problem1

مشکل over-translation است برای آن یک معماری جدید پیشنهاد شده است به نام GRU-gated attention model که بعد از هر گام رمزگشایی و قبل از اینکه به لایه بعد برسیم یک گیت قرار می دهیم که توجه گفته شده را به گونه ای دیگر حساب کند.

۱. gating layer:

هدف این لایه بهبود ارائه منبع با توجه به حالت رمزگشای قبلی (s_{j-1}) به منظور محاسبه بازنمایی منبع مرتبط ترجمه است. رابطه ی این لایه به شکل زیر است:

$$\mathbf{H}_j^g = \text{Gate}(\mathbf{H}, \mathbf{s}_{j-1})$$

در رابطه بالا Gate باید بتواند با ارتباط پیچیده بین منبع ورودی و ترجمه کار کند و به راحتی رابطه معنایی و جریان اطلاعات بین ورودی و ترجمه را کنترل کند.

به جای استفاده از مکانیزم gating کانولوشنی در این روش از کل ساختار GRU برای انجام این وظیفه استفاده میکنیم. که در این حالت روابط بهبود حالت رمزگشای قبلی (s_{j-1}) عبارت است از:

$$\begin{aligned}
\mathbf{z}_{ji} &= \sigma(W_z\mathbf{s}_{j-1} + U_z\mathbf{h}_i + b_z) \\
\mathbf{r}_{ji} &= \sigma(W_r\mathbf{s}_{j-1} + U_r\mathbf{h}_i + b_r) \\
\overline{h}_{ji} &= \tanh(W\mathbf{s}_{j-1} + U\left[\mathbf{r}_{ji} \odot \mathbf{h}_i\right] + b) \\
\mathbf{h}_{ji}^g &= (1 - \mathbf{z}_{ji}) \odot \mathbf{h}_i + \mathbf{z}_{ji} \odot \overline{h}_{ji}
\end{aligned}$$

که σ همان sigmoid و ⊙ ضرب عنصری است و دو گیت r_{ji} و z_{ji} معیاری از درجه معنادار بودن و ارتباط معنایی بین ورودی و ترجمه هستند.

۲. attention layer:

این لایه با لایه اصلی در مکانیزم توجه یکسان است و رابطه ی آن عبارت است از:

$$\mathbf{c}_j = \text{Att}(\mathbf{H}_j^g, \mathbf{s}_{j-1})$$

به هر حال به جای پرداختن به بازنمایی منبع H، این لایه به بهبود گیت H^g وابسته است. همانطور که مشخص است H^g در طول decoding پویا است. باید به این نکته توجه کرد که Gate(.) یک RNN چند مرحله ای نیست و یک تابع ساده‌ی ترکیبی و یا یک RNN تک مرحله ای است پس در نتیجه از لحاظ محاسباتی بسیار موثر است. برای آموزش این مدل همانند قبل هدف بهینه سازی را ماکسیمم کردن مقدار log-likelihood در دادگان آموزشی میگذاریم و با استفاده از بهینه ساز گرادیان کاهشی تصادفی، پارامترهای مدل را بهبود میدهیم.

حال میتوان با استفاده کردن از حالت decoder قبلی به عنوان تاریخچه و منبع بازنمایی ورودی جدید مدلی تحت عنوان GAtt-Inv طراحی کرد که روابط آن برابر است با:

$$\mathbf{c}_j = \text{GAtt-Inv}(\mathbf{H}, \mathbf{s}_{j-1})$$

$$\mathbf{c}_j = \text{Att}(\mathbf{H}_j^{g\prime}, \mathbf{s}_{j-1}) \quad \mathbf{H}_j^{g\prime} = \text{Gate}(\mathbf{s}_{j-1}, \mathbf{H})$$

جدول نتیجه برای مقایسه به شرح زیر است:

System	SAER	AER
Tu et al. [2016]	64.25	50.50
RNNSearch	64.10	50.83
GAtt	**56.19**	**43.53**
GAtt-Inv	58.29	43.60

Table 4: SAER and AER scores of word alignments deduced by different neural systems. The lower the score, the better the alignment quality.

System	1-gram	2-gram	3-gram	4-gram
Reference	12.94	1.80	0.93	1.29
RNNSearch	19.12	5.26	3.27	2.97
GAtt	18.09	4.11	2.50	2.46
GAtt-Inv	16.79	3.39	1.99	1.94

Table 5: N-GRR scores of different systems on all test sets with N ranges from 1 to 4. The lower the score, the better the system deals with the over-translation problem.

منبع :

https://arxiv.org/pdf/1704.08430.pdf

Problem 2

توجه نرم :

برای محاسبه context vector از جمع وزن دار حالات پنهان استفاده میکنیم (در واقع attention score هایی که محاسبه کردیم)

توجه سخت:

در این حالت به جای جمع وزن دار حالات ، فقط از یک حالت پنهان استفاده می کنیم به طوری که attention score های هر حالت را محاسبه کرده و از بینشان یکی را با یک تابع انتخاب میکنیم(غیر max چون مشتق پذیر نیست)تا حالتی که attention score ای که بالاتر است را انتخاب می کنیم . خلاصه این دو روش در شکل زیر آمده است .

Soft Attention	*Hard Attention*
Attention score is used as weights in the weighted average context vector calculation. This is a differentiable function.	*Attention score is used as the probability of the i-th location getting selected. We could use a simple argmax to make the selection, but it is not differentiable and so complex techniques are employed.*
$\mathbb{E}_{p(s_t\|a)}[\hat{\mathbf{z}}_t] = \sum_{i=1}^{L} \alpha_{t,i}\mathbf{a}_i$	$\hat{\mathbf{z}}_t = \sum_i s_{t,i}\mathbf{a}_i$ $p(s_{t,i} = 1 \mid s_{j<t}, \mathbf{a}) = \alpha_{t,i}$ $\tilde{s_t^n} \sim \text{Multinoulli}_L(\{\alpha_i^n\})$

Problem3

A

می توان گفت که single attention layer از آنجا که صرفا یک رابطه خطی طور است در نهایت ظرفیت کمی برای یادگیری دارد و افزایش تعداد لایه ها باعث افزایش ظرفیت یادگیری می شود اما از طرف دیگر، همانطور که در این مقاله ذکر شده است افزایش تعداد لایه ها همیشه لزوما باعث افزایش ظرفیت یادگیری شبکه نمی شود و چنانچه نسبت بین عمق و پهنای شبکه از یک آستانه بیشتر شود، در این صورت حتی ممکن است شبکه با عمق کمتر عملکرد بهتری داشته باشد . از طرف دیگر می توان گفت ممکن است با افزایش تعداد لایه ها تمرکز بر روی داده خاص بیشتر شود و عملکرد مناسبی روی train داشته باشیم ولی در حالی که عملکرد خوبی بر test set نداریم و عملا overfit شده ایم

Theorem 2. *For $y_p^{i,L,d_x,H,\Theta}$ as defined in theorem 1, if $L > \log_3(d_x)$, then the following holds almost everywhere in the network's learned parameter space,* i.e. *for all values of the weight matrices (represented by Θ) but a set of Lebesgue measure zero:*

$$\frac{1}{2}d_x \cdot L + b_1 + b_2 \leq \log_3\left(sep(y_p^{i,L,d_x,H,\Theta})\right) \leq 2d_x \cdot L + c_1 + c_2 \qquad (7)$$

with corrections on the order of L: $b_1 = -L\left(\frac{H}{2}+1\right)$, $c_1 = L$, and on the order of $d_x \log_3(d_x)$: $b_2 = -d_x\left(1 + \frac{1}{2}\log_3\left(\frac{d_x - H}{2}\right)\right)$, $c_2 = -2d_x \cdot \log_3 d_x/2\sqrt{2e} + \log_3 d_x$.

B

خیر نمی شود ، از انجا که برای هر key یک value داریم پس ابعاد ماتریس key و value دارای تعداد سطر یکسان هستند ، حال آنکه با ضرب بردار query با ابعاد متفاوت نسبت به key در ماتریس key نمیتوان چنین پیش شرطی را ارضا کرد.

C

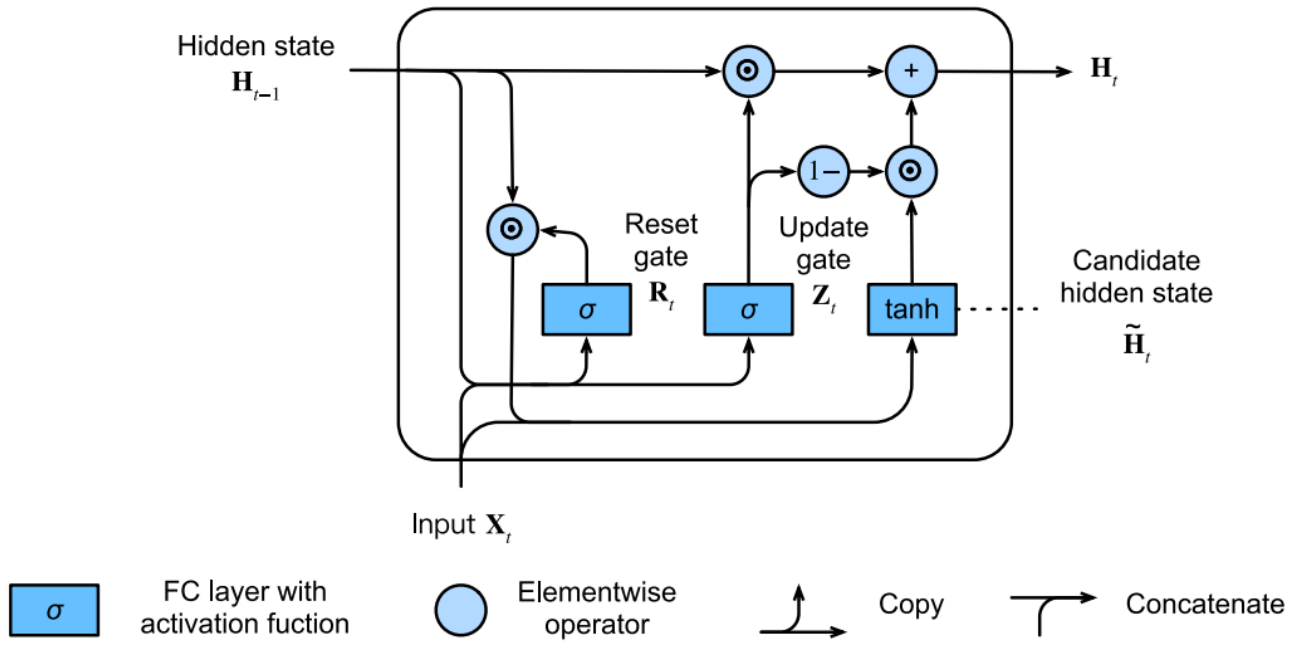
استفاده از ضرب در تابع امتیاز دهی بین key ها و query ها باعث می شود که در واقع بتوانیم مقدار نزدیک بود و همجهت بودن key ها را با query مد نظر بسنجیم ، به طوری که اگر این دو همجهت و هم علامت بودند خروجی شان مقدار مثبت و نسبتا بزرگی به نسبت ضرب بقیه key ها با query مد نظر است و این باعث میشود خروجی softmax هم عدد بالاتری باشد و وزن بالاتری به value متناظر با key ای که امتیاز بالاتری اخذ کرده داده شود. حال آنکه جمع کردن توان محاسباتی ضرب برای این چنین کاربردی را ندارد و صرفا یک مقدار ثابت (query) را با key های مختلف جمع کرده و به softmax تحویل می دهد که این محاسبات در نهایت نمیتوانند وزن مناسبی به value متناظر با key نزدیک تر به query تخصیص دهد.

D

می خواهیم خروجی هر مرحله فقط به یکی از ورودی های مرحله قبل وابسته باشد .

به دو واژه یکی و قبل دقت کنید، در واقع چون خروجی به مرحله ی قبل وابسته است نه مرحله ی فعلی باعث می شود که مقدار **Zt در Update gete برابر یک** تنظیم شود ، چرا که با توجه به فرمول update gate که در کتاب آمده با یک شدن این مقدار در واقع ما مقدار state قدیمی را بر می گردانیم (صفر شدن این مقدار باعث می شود مقدار state فعلی برگردانده شود)

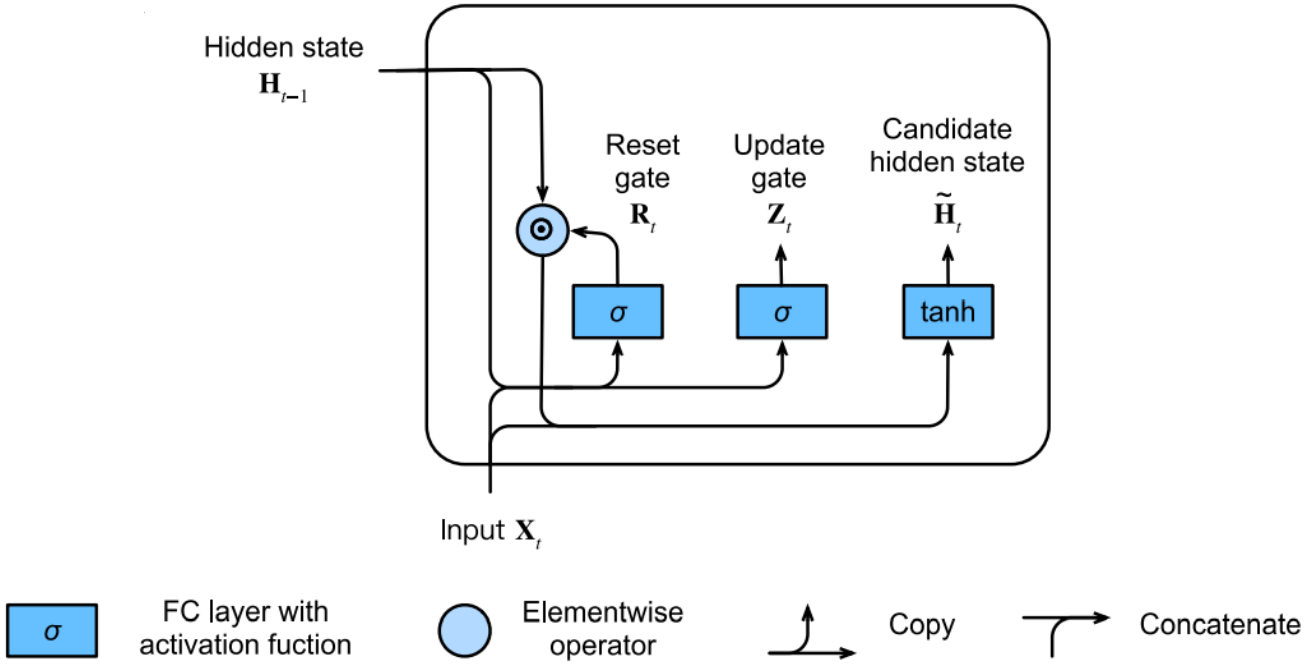
$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t. \tag{9.1.4}$$



در مورد واژه ی یکی و reset gate هم می توان گفت از انجا که قصد داریم hidden state فعلی را با hidden state قبل قبلی (با استفاده از update gate که بالاتر توضیح داده شد) جایگزین کنیم و فقط هم به یک ورودی مرحله قبل وابسته باشد مجبوریم مقدار **Rt در Reset gate را صفر** تنظیم کنیم . چرا که همانطور که در فرمول زیر که در کتاب آمده هم اشاره شده

دقت کنیم، که اگر مقدار Rt برابر صفر در نظر گرفته شود در واقع ما فقط با Xt طرف هستیم و چنانچه این مقدار به 1 میل کند ما علاوه بر Xt با H t-1 هم طرفیم (یک RNN تمام عیار!)

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h). \tag{9.1.3}$$



E

می توان گفت که Reset gate برای مدیریت short term dependencies به کار می رود در حالی که Update gate برای مدیریت long term dependencies

پیاده سازی update gate باعث می شود که همچنان بتوانیم از state های گذشته ها (و حتی گذشته های دور) استفاده کنیم، حال آنکه عدم پیاده سازی Reset gate باعث می شود که در مورد short term ها مدیریت لازم را اعمال کنیم و در واقع در خروجی logit هر حالتی ، صرفا Xt همان حالت و لحظه به کار گرفته شود (از RNN بودن میافتد) چرا که شبکه دیگر عملکرد مناسبی نسبت به short term ها و همسایه های نزدیک ندارد .

Problem4

GRU+DROP	GRU	LSTM+DROP	LSTM	RNN + DROP	Simple RNN	Model
60.1	59.5	59.8	61.2	43.3	51.7	accuracy

مشاهده می شود که در مورد Simple RNN و LSTM افزایش لایه ها باعث کاهش دقت می شود ، حتی با وجود Drop out در مورد GRU اما اضافه کردن لایه ها و لایه DROP OUT باعث افزایش accuracy می شود.

بهترین عملکرد مربوط به LSTM بوده و پس از آن GRU با لایه های زیاد و DROP OUT عمکرد مناسب تری داشته.

بدترین عملکرد هم مربوط به RNN+ drop می باشد.