-1

با autoregressive factorization (فاکتوراسیون خود رگرسیون) ، مدل سازی زبان را می توان به مدل کردن توزیع شرطی توکن بعدی (یعنی) x با توجه به متن c (context) تقلیل داد. اگرچه ممکن است ادعا شود که یک زبان طبیعی به دلیل ترکیب بندی خود تعداد بی شماری از متن‌ها را مجاز می داند ، ما با در نظر گرفتن یک مجموعه محدود از متن‌های ممکن ، تحلیل خود را ادامه می دهیم. محدودیت زبان طبیعی نتیجه گیری ما را تحت تأثیر قرار نمی دهد.

ما ما یک زبان طبیعی را به عنوان یک مجموعه متناهی از زوج‌های یک متن و توزیع شرطی توکن بعدی آن در نظر می گیریم $\mathcal{L} = \{(c_1, P^*(X|c_1)), \cdots, (c_N, P^*(X|c_N))\}$
که N تعداد متن‌های ممکن است. برای احتساب خطا و انعطاف‌پذیری زبان‌های طبیعی فرض می‌کنیم $P^* > 0$ همه‌جا برقرار است. $\{x_1, x_2, \ldots, x_M\}$ یک مجموعه شامل M توکن ممکن در زبان L را نشان می‌دهد. هدف مدل زبانی این است که توزیع $p_\theta(X|c)$ که توسط θ پارامتر شده با توزیع واقعی داده تطبیق یابد. زبان طبیعی L را داریم. آیا یک پارامتر θ وجود دارد که برای تمام cهای درون L داشته باشیم : $P_\theta(X|c) = P^*(X|c)$
برای ساخت این مدل زبانی مدل‌های مبتنی بر Softmax را بررسی می‌کنیم.

:Softmax

اکثر مدل‌های زبانی پارامتریک از یک تابع Softmax استفاده می کنند که بر روی یک بردار متن context vector حالت پنهان (یا hidden statate) h_c و یک کلمه تعبیه شده word embedding w_x برای تعریف توزیع شرطی $p_\theta(X|c)$ کار می کند. به طور خاص ، توزیع مدل معمولاً به صورت زیر نوشته می شود:

$$P_\theta(x|c) = \frac{\exp \mathbf{h}_c^\top \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}_c^\top \mathbf{w}_{x'}}$$

که h_c تابعی از c و w_x تابعی از x است. هر دو تابع توسط θ پارامتر شده‌اند. هم context vector (بردار متن) h_c و هم word embedding (تعبیه کلمه) w_x دارای یک بعد یکسان d هستند. حاصل ضرب داخلی $h^\top_c w_x$ را logit می‌نامند.

برای کمک به بحث درباره‌ی expressiveness مدل softmax سه معیار زیر را تعریف می‌کنیم:

$$\mathbf{H}_\theta = \begin{bmatrix} \mathbf{h}_{c_1}^\top \\ \mathbf{h}_{c_2}^\top \\ \cdots \\ \mathbf{h}_{c_N}^\top \end{bmatrix};\ \mathbf{W}_\theta = \begin{bmatrix} \mathbf{w}_{x_1}^\top \\ \mathbf{w}_{x_2}^\top \\ \cdots \\ \mathbf{w}_{x_M}^\top \end{bmatrix};\ \mathbf{A} = \begin{bmatrix} \log P^*(x_1|c_1), & \log P^*(x_2|c_1) & \cdots & \log P^*(x_M|c_1) \\ \log P^*(x_1|c_2), & \log P^*(x_2|c_2) & \cdots & \log P^*(x_M|c_2) \\ \vdots & \vdots & \ddots & \vdots \\ \log P^*(x_1|c_N), & \log P^*(x_2|c_N) & \cdots & \log P^*(x_M|c_N) \end{bmatrix}$$

$$\mathbf{H}_\theta \in \mathbb{R}^{N\times d}, \mathbf{W}_\theta \in \mathbb{R}^{M\times d}, \mathbf{A} \in \mathbb{R}^{N\times M}.$$

سطرهای H_θ، W_θ و A به ترتیب با context vectorها ، word embeddingها و لگاریتم احتمال توزیع داده های واقعی مطابقت دارند. H_θ به عنوان شبکه های عصبی عمیق مانند یک شبکه recurrent پیاده‌سازی می شود ، در حالی که W_θ به عنوان lookup embedding نمونه سازی شده است.

علاوه بر این یک مجموعه ازماتریس‌ها را مشخص می‌کنیم که با شیفت سطری row-wise shift بر روی A ایجاد شده‌اند:

$$F(\mathbf{A}) = \{\mathbf{A} + \mathbf{\Lambda J}_{N,M} | \mathbf{\Lambda} \text{ is diagonal and } \mathbf{\Lambda} \in \mathbb{R}^{N \times N}\},$$

و $J_{N,M}$ یک ماتریس N×N است که تمام مقادیرش یک هستند.

اساسا عملیات شیفت سطری row-wise shift یک عدد حقیقی دلخواه را به هر سطر A اضافه می‌کند. بنابراین F(A) یک مجموعه نامتناهی است.

مجموعه‌ی F(A) دارای دو ویژگی مهم هستند که کلید تحلیل ما می‌باشند.

ویژگی 1: برای هر ماتریس A′ ، A′ є F(A) اگر و تنها اگر softmax(A′) = P* . به عبارت دیگر F(A) مجموعه‌ی تمام logitهای ممکن که با توزیع واقعی داده‌ها تطابق دارند را تعریف می‌کند.

ویژگی 2: برای هر A1 ≠ A2 є F(A) داریم : | rank(A2)-rank(A1) |<= 1 . به عبارت دیگر تمام ماتریس‌ها در F(A) دارای مرتبه(rank)های مشابه هستند با حداکثر اختلاف 1:

بر اساس ویژگی 1 اصل (lemma) زیر را داریم:

lemma1: با توجه به پارامتر θ مدل $H_\theta W^T_\theta$ є F(A) اگر و تنها اگر برای تمام cها در L داشته باشیم:

$$P_\theta(X|c) = P^*(X|c)$$

حال سوال این است که آیا پارامتر θ و A′ є F(A) وجود دارد بطوریکه $H_\theta W^T_\theta = A'$

این اساسا یک مسئله matrix factorization است.

ما می‌خواهیم مدل ماتریس‌های H_θ و W_θ را طوری یاد بگیرد که قادر به factorize کردن ماتریسی مانند A′ є F(A) باشد. ابتدا توجه داشته‌باشید که برای اینکه یک factorization معتبر وجود داشته باشد مرتبه (rank) $H_\theta W^T_\theta$ باید حداقل برابر مرتبه (rank) A′ باشد. علاوه بر این چون $H_\theta \epsilon R^{N\times d}$ و $W_\theta \epsilon R^{M\times d}$ بنابراین مرتبه (rank) $H_\theta W^T_\theta$ با اندازه embedding (تعبیه) d محدود شده است. در نتیجه اگر d >= rank(A′) یک تقریب‌زن جهانی (universal approximator) از نظر تئوری می‌تواند A′ را بازیابی کند. با این حال اگر d < rank(A′) دیگر مهم نیست خانواده تابع U چقدر expressive است و هیچ (H_θ و W_θ) نمی‌تواند از نظر تئوری A′ را بازیابی کند.

استدلال فوق را به صورت زیر خلاصه می‌کنیم.

قضیه 1: اگر خانواده توابع U که یک تقریب زن جهانی (universal approximator) را داشته باشیم یک پارامتر مانند θ وجود دارد بطوریکه برای تمام cهای در L $P_\theta(X|c) = P^*(X|c)$ برقرار است اگر و تنها اگر $d > \min_{A' \epsilon F(A)} \mathrm{rank}(A')$

با ترکیب ویژگی 2 و قضیه 1 اکنون می‌توانیم مشکل softmax bottleneck را به صورت رسمی بیان کنیم.

نتیجه 1 (softmax bottleneck): اگر d < rank(A’) -1 برای هر خانواده تابع U و پارامتر θ حداقل یک متن (context) وجود دارد به طوری که $P_\theta(X|c) \neq P^*(X|c).$

نتیجه‌ی فوق نشان می‌دهد که هنگامی که بعد d خیلی کوچک باشد softmax توان بیان (express) توزیع واقعی داده‌ها را ندارد. بدیهی است که این نتیجه‌گیری به یک زبان متناهی L محدود نمی‌شود. وقتی L نامتناهی است همیشه می‌توان یک زیرمجموعه‌ی متناهی از آن را انتخاب کرد و مشکل softmax bottleneck همچنان وجود دارد. در مرحله‌ی بعد با این فرض که زبان‌های طبیعی دارای مرتبه (rank) بالایی هستند نشان می‌دهیم که چرا softmax bottleneck یک مشکل است.

فرض می‌کنیم برای زبان طبیعی L، ماتریس (log probability) A یک ماتریس با مرتبه‌ی بالا (high rank) باشد. اثبات این فرض اگرچه ممکن است به دلیل اینکه ما به توزیع واقعی داده‌ها دسترسی نداریم سخت به نظر برسد اما با استدلال‌های شهودی و مشاهدات تجربی توصیه شده است.

زبان طبیعی به شدت وابسته به متن (context) می‌باشد به عنوان مثال توکن ”north“ با احتمال زیاد در یک مقاله خبری درباره‌ی سیاست قبل از ”korea“ یا ”Korean“ می‌آید که بعید است در در یک کتاب درسی درباره‌ی تاریخ ایالات متحده باشد. ما فرض می‌کنیم چنین وابستگی ظریفی به متن باید منجر به یک ماتریس با مرتبه‌ی بالا (high rank) شود.

اگر A دارای مرتبه پایین (low rank) باشد به این معنی است که انسان به تعداد محدودی در حد چند صد مبنا احتیاج دارد و تمام معانی نحوی را می‌توان با (به طور بالقوه) نفی و میانگین وزن‌دار این مبانی ایجاد کرد. با این وجود یافتن مفهومی طبیعی در زبانشناسی و علوم طبیعی که با چنین مبناهایی مطابقت داشته باشد دشوار است که وجود چنین مبناهایی را زیر سوال می‌برد.

به صورت تجربی نشان داده شده‌است که مدل‌های زبانی مرتبه‌ی بالا (high rank) از مدل‌های مرتبه پایین (low rank) مرسوم عملکرد بهتری دارند. یادگیری مدل‌های زبانی مرتبه‌ی بالا (high rank) مهم است.

با توجه به این فرض که زبان‌های طبیعی زبان‌های طبیعی مرتبه‌ی بالا (high rank) هستند بدیهی است که softmax bottleneck ، expressiveness مدل را محدود می‌کند. در عمل embedding (تعبیه) d معمولا در مقیاس 10^2 تنظیم می‌شود در حالی که مرتبه‌ی (rank) A احتمالا می‌تواند به اندازه‌ی M (در مقیاس 10^5) باشد که به شدت بزرگتر از d هست.

Softmax در بهترین حالت یادگیری تقریبی با مرتبه پایین (low rank) از A است و تجربه‌ها نشان می‌دهد که چنین تقریبی توانایی مدل‌سازی وابسته به متن (context) را از نظر کمی و کیفی ندارد.

راه‌حل آسان (easy fix):

با رخ دادن softmax bottleneck بلافاصله برخی حل‌های آسان به ذهن می‌رسد. اول اینکه می‌توان از یک مدل غیرپارامتریک مانند Ngram استفاده کرد. مدل‌های Ngram توسط هیچ فرم پارامتری محدود نمی‌شوند، بنابراین با داشتن پارامترهای کافی می‌توانند هر زبان طبیعی را به صورت عمومی (universal) تقریب بزنند. دوم این که

امکان افزایش بعد d (مثلا تا حد برابر شدن با M) وجود دارد تا مدل بتواند یک ماتریس مرتبه‌ی بالا (high rank) A را بیان (express) کند.

با استفاده از این دو روش در مقایسه با استفاده از softmax با ابعاد پایین تعداد پارامترها به صورت چشمگیری افزایش میابد. به صورت دقیق‌تر یک مدل Ngram برای بیان (express) A نیاز به N×M پارامتر دارد که N به طور بالاقوه نامتناهی است. به همین ترتیب یک softmax با ابعاد بالا به M×M پارامتر برای تعبیه کلمه (word embedding) نیاز دارد. افزایش پارامترهای مدل به سادگی منجر به overfit شدن می‌شود. از روش Back-off برای کاهش overfit استفاده شده است. علاوه بر این از آنجا که مدل‌های یادگیری عمیق با جست‌وجوی گسترده‌ی پارامترها tune (تنظیم) می‌شوند افزایش ابعاد d فراتر از چندصد واحد به صرفه نیست.

بدیهی است که بین expressiveness و generalization (تعمیم) در مدل‌سازی زبان یک trade off وجود دارد. افزایش ساده‌لوحانه expressiveness منجر به آسیب زدن به generalization می‌شود. حال به معرفی یک روش می‌پردازیم که expressiveness را بدون نیاز به افزایش بیش از حد پارامترها بالا ببرد.

MIXTURE of Softmaxes (یک مدل زبانی مرتبه بالا (high rank)): MoS

MoS توزیع شرطی را به صورت زیر محاسبه می‌کند.

$$P_\theta(x|c) = \sum_{k=1}^{K} \pi_{c,k} \frac{\exp \mathbf{h}_{c,k}^\top \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}_{c,k}^\top \mathbf{w}_{x'}}; \text{ s.t. } \sum_{k=1}^{K} \pi_{c,k} = 1$$

که $\pi_{c,k}$ وزن پیشین (prior) یا ترکیبی (mixture) kامین مولفه است و $h_{c,k}$ ، kامین بردار متن (context vector) است ک با متن (context) c مرتبط است. به عبارت دیگر، MoS توزیع k عدد softmax را محاسبه می‌کند و از میانگین وزن‌دار آن‌ها به عنوان توزیع احتمال توکن بعدی استفاده می‌کند. ابتدا یک دسته از لایه‌های recurrent را در بالای X اعمال می‌کنیم تا یک ترتیب از حالتهای پنهان (hidden states) به دست آوریم.

$$\pi_{c_t,k} = \frac{\exp \mathbf{w}_{\pi,k}^\top \mathbf{g}_t}{\sum_{k'=1}^{K} \exp \mathbf{w}_{\pi,k'}^\top \mathbf{g}_t} \qquad \mathbf{h}_{c_t,k} = \tanh(\mathbf{W}_{h,k}\mathbf{g}_t)$$

که $w_{\pi,k}$ و $W_{h,k}$ پارامترهای مدل هستند.

این روش ساده و پیاده‌سازی آن آسان است و مزایای زیر را دارد:

1. بهتر شدن expressiveness در مقایسه با softmax: با همان بعد d از نظر تئوری بیشتر از (یا حداقل به همان اندازه) expressive است. این را می‌توان با این واقعیت نشان داد که MoS با K=1 به softmax تقلیل میابد. مهمتر از همه MoS بهتر A را تقریب می‌زند با :

$$\hat{\mathbf{A}}_{\text{MoS}} = \log \sum_{k=1}^{K} \mathbf{\Pi}_k \exp(\mathbf{H}_{\theta,k}\mathbf{W}_\theta^\top)$$

که $\mathbf{\Pi}_k$ یک ماتریس قطری N×N است که اعضای آن $\pi_{c,k}$ است.

از آنجا که $\hat{A}_{MoS}$ یک تابع غیر خطی (log-sum-exp) از بردارهای متن (context vectors) و word embeddingها است می‌تواند مرتبه‌ی بالای (high rank) دلخواهی داشته باشد در نتیجه MoS مانند softmax از محدودیت مرتبه (rank) رنج نمی‌برد.

2. بهتر شدن generalization (در مقایسه با Ngram): مدل‌های Ngram و softmax با ابعاد بالا expressiveness را بهبود می‌بخشند اما به خوبی تعمیم نمی‌یابند. در مقابل MoS به دلایل زیر مشکلات تعمیم را ندارند.

MoS فرایند مولد زیر را تعریف می‌کند:

یک متغیر گسسته تهفته k ابتدا از {1,2,...,K} نمونه‌برداری می‌شود و سپس توکن بعدی بر اساس مولفه‌ی softmax Kام نمونه‌برداری می‌شود. با این کار ما یک bias استقرایی را معرفی می کنیم که توکن بعدی بر اساس یک تصمیم گسسته نهفته تولید می شود که اغلب در مدل‌سازی زبان‌ها ایمن است. دوم، از آنجا که $\hat{A}_{MoS}$ توسط یک تابع غیرخطی تعریف می‌شود و با مرتبه bottleneck (rank) محدود نمی‌شود در عمل می‌توان d را کاهش داد تا افزایش پارامترهایی که توسط ساختار mixture آورده می‌شود جبران شود. در نتیجه MoS در مقایسه با softmax دارای اندازه‌ی مدل مشابهی است و بنابراین مستعد overfit شدن نیست.

MIXTURE of CONTEXTS

یک رویکرد دیگر این است که بردارهای متن (context vectors) را قبل از استفاده در softmax ترکیب کنیم نه ترکیب کردن احتمال‌ها بعد از softmax مانند MoS. توزیع شرطی آن به صورت زیر پارامتر شده است

$$P_\theta(x|c) = \frac{\exp\left(\sum_{k=1}^{K} \pi_{c,k}\mathbf{h}_{c,k}\right)^\top \mathbf{w}_x}{\sum_{x'} \exp\left(\sum_{k=1}^{K} \pi_{c,k}\mathbf{h}_{c,k}\right)^\top \mathbf{w}_{x'}} = \frac{\exp\left(\sum_{k=1}^{K} \pi_{c,k}\mathbf{h}_{c,k}^\top \mathbf{w}_x\right)}{\sum_{x'} \exp\left(\sum_{k=1}^{K} \pi_{c,k}\mathbf{h}_{c,k}^\top \mathbf{w}_{x'}\right)},$$

که در آن $h_{c,k}$ و $\pi_{c,k}$ پارامترهایی مشابه MoS دارند. علارغم شباهت ظاهری آن با MoS این مدل که از آن با عنوان ترکیبی از متن‌ها MoC (mixture of contexts) یاد می‌کنیم در واقع مانند softmax از مشکل محدودیت مرتبه (rank) رنج می‌برد. این واقعیت با تعریف $\mathbf{h}'_c = \sum_{k=1}^{K} \pi_{c,k}\mathbf{h}_{c,k}$ که پارامترسازی فوق را به

$P_\theta(x|c) = \frac{\exp \mathbf{h}'^\top_c \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}'^\top_c \mathbf{w}_{x'}}$ تبدیل می‌کند به راحتی دیده می‌شود. توجه داشته باشید که این با پارامترسازی Softmax هم ارز است. بنابراین انجام عمل ترکیب (mixture) در فضای ویژگی‌ها تنها expressiveness را بیشتر می‌کند اما این واقعیت را تغییر نمی‌دهد که مرتبه $H_\theta W_\theta^\top$ (rank) توسط embedding (تعبیه) d محدود شده است.

نتیجه‌گیری: تحت چارچوب matrix factorization ، expressiveness مدل‌های زبانی مبتنی بر softmax با ابعاد embedding کلمات محدود شده است. که softmax bottleneck نامیده می‌شود. مدل MoS پیشنهادی

expressiveness را نسبت به softmax بهبود می‌بخشد و در عین حال در مقایسه با مدل‌های غیرپارامتریک و افزایش ساده‌لوحانه ابعاد embedding کلمه از overfit شدن جلوگیری می‌کند.

2- روش اول: محاسبه مشتق تک تک پارامترها به صورت مستقیم.

$$y = \omega^T \max\{0, W^T x + c\} + b$$

$$J(\theta) = \frac{1}{N} \sum (y^* - y)^2$$

$$y = \omega_1 \max\{0, W_{11} x_1 + W_{21} x_2 + c_1\} + \omega_2 \max\{0, W_{12} x_1 + W_{22} x_2 + c_2\} + b$$

$$\frac{\partial J}{\partial W_{11}} = \begin{cases} 0 & W_{11} x_1 + W_{21} x_2 + c_1 < 0 \\ -\frac{2}{N} \omega_1 x_1 \sum (y^* - y) & ow \end{cases}$$

$$\frac{\partial J}{\partial W_{21}} = \begin{cases} 0 & W_{11} x_1 + W_{21} x_2 + c_1 < 0 \\ \frac{-2 \omega_1 x_2}{N} \sum (y^* - y) & ow \end{cases}$$

$$\frac{\partial J}{\partial c_1} = \begin{cases} 0 & W_{11} x_1 + W_{21} x_2 + c_1 < 0 \\ \frac{-2}{N} \omega_1 \sum (y^* - y) & ow \end{cases}$$

$$\frac{\partial J}{\partial \omega_1} = \begin{cases} 0 & W_{11} x_1 + W_{21} x_2 + c_1 < 0 \\ \frac{-2}{N} (W_{11} x_1 + W_{21} x_2 + c_1) \sum (y^* - y) & ow \end{cases}$$

$$\frac{\partial J}{\partial w_{12}} = \begin{cases} w_{12}x_1 + w_{22}x_2 + c_2 < 0 & 0 \\ \text{ow} & \frac{-2w_2x_1}{N}\sum(y^* - y) \end{cases}$$

$$\frac{\partial J}{\partial w_{22}} = \begin{cases} w_{12}x_1 + w_{22}x_2 + c_2 < 0 & 0 \\ \text{ow} & \frac{-2w_2x_2}{N}\sum(y^* - y) \end{cases}$$

$$\frac{\partial J}{\partial c_2} = \begin{cases} w_{12}x_1 + w_{22}x_2 + c_2 < 0 & 0 \\ \text{ow} & \frac{-2w_2}{N}\sum(y^* - y) \end{cases}$$

$$\frac{\partial J}{\partial w_2} = \begin{cases} w_{12}x_1 + w_{22}x_2 + c_2 < 0 & 0 \\ \text{ow} & (w_{12}x_1 + w_{22}x_2 + c_2)\ \frac{-2}{N}\sum(y^* - y) \end{cases}$$

$$\frac{\partial J}{\partial b} = \frac{-2}{N}\sum(y^* - y)$$

روش دوم : استفاده از backpropagation

$$J(\theta) = \frac{1}{N} \sum (y^* - y)^2$$

$$y = \omega_1 h_1 + \omega_2 h_2 + b \qquad h_1 = \max\{0, W_{11} x_1 + W_{21} x_2 + c_1\}$$

$$h_2 = \max\{0, W_{12} x_1 + W_{22} x_2 + c_2\}$$

$$\frac{\partial J}{\partial y} = \frac{-2}{N} \sum (y^* - y)$$

$$\frac{\partial J}{\partial \omega_1} = \frac{\partial J}{\partial y} \times \frac{\partial y}{\partial \omega_1} = \frac{-2}{N} \sum (y^* - y)\, h_1 =$$

$$\frac{-2}{N} \sum (y^* - y) \max\{0, W_{11} x_1 + W_{21} x_2 + c_1\}$$

$$\frac{\partial J}{\partial \omega_2} = \frac{\partial J}{\partial y} \times \frac{\partial y}{\partial \omega_2} = \frac{-2}{N} \sum (y^* - y)\, h_2 =$$

$$\frac{-2}{N} \sum (y^* - y) \max\{0, W_{11} x_1 + W_{21} x_2 + c_1\}$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial y} \times \frac{\partial y}{\partial b} = \frac{-2}{N} \sum (y^* - y) \times 1 = \frac{-2}{N} \sum (y^* - y)$$

$$\frac{\partial J}{\partial h_1} = \frac{\partial J}{\partial y} \times \frac{\partial y}{\partial h_1} = \frac{-2}{N} \sum (y - y^*)\, \omega_1$$

$$\frac{\partial J}{\partial h_2} = \frac{\partial J}{\partial y} \times \frac{\partial y}{\partial h_2} = \frac{-2}{N} \sum (y^* - y)\, \omega_2$$

$$\frac{\partial J}{\partial W_{11}} = \frac{\partial J}{\partial h_1} \times \frac{\partial h_1}{\partial W_{11}} = \begin{cases} \frac{-2}{N} \sum (y^* - y)\, w_1 x_1 & 0 < W_{11}x_1 + W_{21}x_2 + c_1 \\ 0 & \text{OW} \end{cases}$$

$$\frac{\partial J}{\partial W_{21}} = \frac{\partial J}{\partial h_1} \times \frac{\partial h_1}{\partial W_{21}} = \begin{cases} \frac{-2}{N} \sum (y^* - y)\, w_1 x_2 & 0 < W_{11}x_1 + W_{21}x_2 + c_1 \\ 0 & \text{OW} \end{cases}$$

$$\frac{\partial J}{\partial c_1} = \frac{\partial J}{\partial h_1} \times \frac{\partial h_1}{\partial c_1} = \begin{cases} \frac{-2}{N} \sum (y^* - y)\, w_1 & 0 < W_{11}x_1 + W_{21}x_2 + c_1 \\ 0 & \text{OW} \end{cases}$$

$$\frac{\partial J}{\partial W_{12}} = \frac{\partial J}{\partial h_2} \times \frac{\partial h_2}{\partial W_{12}} = \begin{cases} \frac{-2}{N} \sum (y^* - y)\, w_2 x_1 & 0 < W_{12}x_1 + W_{22}x_2 + c_2 \\ 0 & \text{OW} \end{cases}$$

$$\frac{\partial J}{\partial W_{22}} = \frac{\partial J}{\partial h_2} \times \frac{\partial h_2}{\partial W_{22}} = \begin{cases} \frac{-2}{N} \sum (y^* - y)\, w_2 x_2 & 0 < W_{12}x_1 + W_{22}x_2 + c_2 \\ 0 & \text{OW} \end{cases}$$

$$\frac{\partial J}{\partial c_2} = \frac{\partial J}{\partial h_2} \times \frac{\partial h_2}{\partial c_2} = \begin{cases} \frac{-2}{N} \sum (y^* - y)\, w_2 & 0 < w_{12}x_1 + w_{22}x_2 + c_2 \\ 0 & \text{o.w.} \end{cases}$$

3-

Sequential API

```
model = Sequential()
###### put your implementations here ######
model.add(tf.keras.layers.Input(shape=data_train[0].shape))
model.add(tf.keras.layers.Flatten())
model.add(Dense(units=30, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
###### put your implementations here ######
```

```
[5] #### put your implementation here ####
    model.compile(optimizer='rmsprop', loss='categorical_crossentropy' , metrics=['accuracy'])
    #### put your implementation here ####
```

همانطور که در کد مشاهده می‌شود غیر از لایه خروجی فقط از یک لایه Fully Connected یا Dense استفاده شده است. optimizer را rmsprop و loss را categorical crossentropy انتخاب کردیم و تابع فعالسازی لایه آخر softmax می‌باشد. از accuracy نیز برای نمایش دقت مدل استفاده کردیم. دقت با فقط یک لایه مخفی Fully Connected بعد از 10 epoch به 88 درصد روی داده train و 87 درصد روی داده‌های validation و test رسیده است. دقت داشته باشید که برای افزایش دقت هم تعداد لایه‌ها را اضافه کردیم و هم فقط تعداد نورون‌های همان

یک لایه را اضافه کردیم که اولی تاثیر چندانی نداشت و دومی درصورت اضافه شدن تعداد بسیاری زیادی نورون کمی تاثیرگذار بود اما منجر به overfit شدن می‌شد.

Functional API

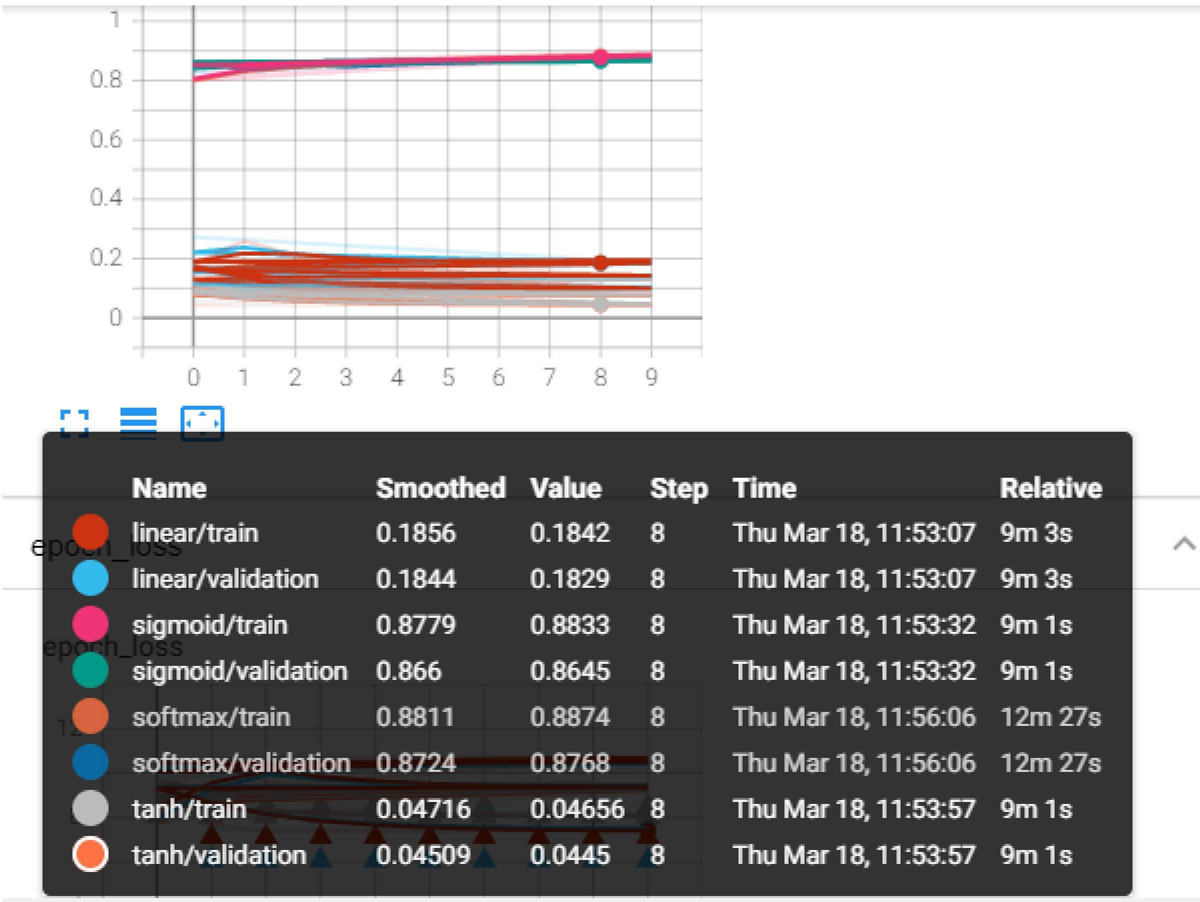
```python
#### put your implementations here ####
model_input = tf.keras.layers.Input(shape=data_train[0].shape)
x = tf.keras.layers.Flatten()(model_input)
x = Dense(units=30, activation='relu')(x)
model_output = Dense(units=10, activation='softmax')(x)


model = tf.keras.Model(model_input, model_output, name="model")
model.summary()
model.compile(optimizer='rmsprop', loss='categorical_crossentropy' , metrics=['accuracy'])
history_f = model.fit(data_train, label_train,
                    epochs=10,
                    batch_size=batch_size,
                    validation_split=0.2, )
#### put your implementations here ####
```

قسمت مربوط به Functional API هیچ توضیح تازه‌ای درباره‌ی تعداد نورون‌ها ، تعداد لایه‌های مخفی، optimizer، loss نداریم و تنها تفاوت این قسمت مربوط به چینش لایه‌ها می‌باشد که آن را در کد و در شکل فوق می‌بینیم.

Custom Training Loop

این قسمت هیچ توضیحی غیر از کد ندارد.

Analyzing the effect of using different activation functions



همانطور که از نمودارهای مربوط به accuracy که از tensorboard گرفته شده است مشخص است دقت برای activation functionهای Linear و tanh به شدت پایین می‌باشد مخصوصا tanh اما دقت sigmoid و softmax از 86 درصد بیشتر می‌باشد که بین این دو دقت softmax بهتر است.