به نام خدا

عنوان:

بخش چهارم تکلیف چهارم شبکه‌های عصبی

استاد:

دکتر منصوری

دانشجو:

محمدعلی مجتهدسلیمانی - ٤٠٣٣٩٠٤٥٠٤

تاریخ:

١٤٠٣/١٠/١٨

Table of Content

بخش اول

چرا Transformer نسبت به RNN در مدل سازی دنباله‌ها بهتر عمل میکند؟

قبل از اینکه به سراغ چرایی بهتر بودن transformer ها برویم بهتر است یک نگاه سطحی به عملکرد هر کدام از این ۲ شبکه عصبی داشته باشیم. RNN ها طراحی شده اند تا دنباله ها را به وسیله نگهداری یک hidden state پردازش کنند. این hidden state در هر گام از دنباله بروز رسانی میشود. این hidden state به عنوان یک حافظه عمل میکند و اطلاعات را از مراحل قبلی نگه داری میکنند تا در پردازش گام فعلی استفاده بکنند. در طرف دیگر Transformer ها با استفاده از مکانیسم self-attention دنباله ها را تغییر میدهند، با این مکانیسم آنها کل دنباله را میتوانند به صورت یکجا و موازی پردازش کنند و نیازی به پردازش ترتیبی ندارند. از اونجایی که در این مکانیسم ترتیبی برای دنباله لحاظ نمیشود از positional encoding استفاده میشود تا اطلاعات مربوط به position هر المان دنباله را نگهداری کند تا ترتیب دنباله ها بهم نخورد.

همانطور که در توضیحات قبلی اشاره کردیم، اصلی ترین مشکل RNN عدم قابلیت موازی بودن است. RNN ها در هر واحد زمان یک المان از دنباله را پردازش میکنند. برای هر المان در هر گام یک hidden state نگهداری و بروز رسانی میشود. این hidden state یک برداری است که اطلاعات گام های قبلی را در خودش نگه میدارد همانطور که در توضیحات بالاتر اشاره کردم. محاسبات hidden state در گام فعلی فقط به تمام hidden state های قبلی وابسته است. این مسئله یک زنجیره ای از وابستگی ها را بین hidden state ها و المان‌های دنباله ها میسازد. همین

مسئله باعث میشود زمان آموزش این شبکه ها بسیار زیاد شود به سبب اینکه یک گلوگاه (bottleneck) در گام فعلی ایجاد میشود که به صورت ترتیبی نیاز هست تمام hidden state ها پردازش شوند.

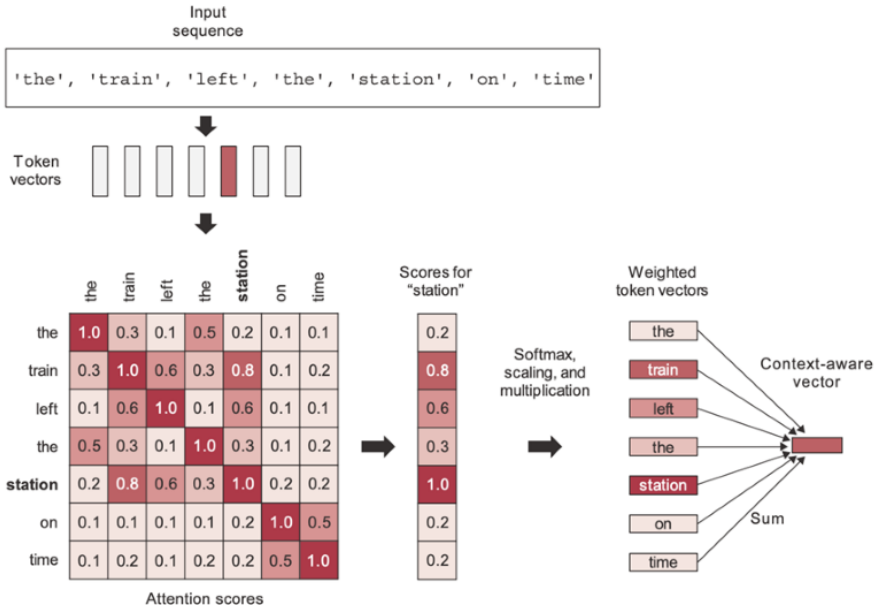
در مقابل Transformer ها به علت استفاده از مکانیسم self-attention امکان موازی سازی دارند زیرا میتوانند به تمام المان های یک دنباله به صورت همزمان دسترسی داشته باشند. به همین دلیل دیگر hidden stateیی وجود نخواهد داشت که بخواهد به گام های بعدی منتقل شود. هر المان به صورت مستقیم میتواند با هر المان دیگر بررسی شود که باعث میشود این زنجیری از وابستگی که در RNN شکل میگرفت از بین برود. این مکانیسم باعث میشود زمان آموزش مدل ها سریعتر شود به همین دلیل ما میتوانیم مدل های بزرگتری بر روی dataset های بزرگ داشته باشیم.

دومین موضوعی که باید به آن اشاره کرد از تفاوت این ۲ شبکه عصبی، توانایی آنها در کار با وابستگی های طولانی یا بلند به اصطلاح (long-range dependencies) گفته میشود به این معنی کدام شبکه عصبی میتواند گذشته های دورتری را به یاد بیاورد. مشکلی که در RNN وجود دارد این است که به علت محو شدن و انفجار گرادیان (Vanishing and Exploding Gradients) دچار مشکل فراموش میشوند در نتیجه این اتفاق شبکه توانایی در بدست آوردن ارتباطات در قسمت های دورتر دنباله را از دست میدهد و به نوعی انگار دچار فراموشی میشود. به عنوان مثال، فهم ضمیر "it" در انگلیسی نیازمند این است که ما بدانیم دقیقا در اوایل جمله کدام اسم (noun) ذکر شده است که در حالا در گام های بعدی به آن اشاره شده است. در حین آموزش RNN از الگوریتم پس انتشار (back propagation) استفاده میکند تا وزن های را تنظیم کند. وقتی دنباله بزرگ بشود این مقدار گرادیان میتواند بسیار کوچک یا بسیار

بزرگ شود. اگر دچار محو شدگی گرادیان شویم باعث میشود گام های ابتدایی دنباله تاثیر بسیار ناچیز و کمی در تنظیم وزن ها داشته باشند، به این معنی که RNN دچار فراموشی نسبت به گام های ابتدایی میشود و آن اطلاعات از بین میروند. اگر گرادیان بسیار بزرگ شود باعث ناپایداری در حین فرآیند یادگیری میشود. (اگر چه که شبکه هایی مانند GRU و LSTM برای حل این مشکلات آمده اند.)

در مقابل Transformer میتوانند دسترسی مستقیم در هر گامی از دنباله داشته باشند. مکانیسم self-attention باعث میشود که بین هر المان یک دنباله ما یک رابطه مستقیم داشته باشیم و اطلاعات مربوط بین این ۲ محاسبه شود. وزن ها اینجا میزان اهمیت هر المان دنباله نسبت به تمام المان های دیگر و خودش محاسبه خواهد شد. به سبب اینکه دیگر پردازش ترتیبی نداریم دیگر خبری از hidden state نداریم. این ویژگی باعث میشود که وابستگی های طولانی بسیار خوب بتوانند محاسبه بشوند و شبکه درک خیلی خوبی از مضمون کلی دنباله داشته باشد. حتی میتوانند ارتباطات پیچیده درون داده ها بسیار خوب متوجه بشوند.

همین امر سبب میشود که در محاسبات این ۲ شبکه تفاوت اساسی ایجاد شود. در RNN ها تعداد محاسبات به صورت خطی با طول دنباله رشد پیدا میکند. یعنی اگر طول دنباله را ۲ برابر کنیم تعداد محاسبات هم ۲ برابر میشود. در مقابل Transformer ها تعداد محاسباتی ثابتی دارند نسبت به طول دنباله که دریافت میکنند. اگر چه که این شبکه ها نیاز به حافظه بیشتری دارند زیرا نیاز هست که وزن های attention ها (امتیاز که نشان میدهد هر المان چه قدر به سایر المان ها توجه دارد) ذخیره شوند. این وزن ها برای هر جفت از المان ها محاسبه میشوند.



این عکس به خوبی محاسبه attention را نشان میدهد که خروجی حاصل یک ماتریس خواهد بود که ابعاد آن برابر با ابعاد خود دنباله است. اعداد بدست آمده همان وزن های attention هستند که نشان میدهد هر المان چه قدر نسبت به خودش و سایر المان ها توجه کند و اهمیت دارد. به عنوان مثال کلمه train با کلمه station وزن بیشتری دارند به این معنا که این ۲ ارتباط بیشتری با همدیگر دارند.

به سبب دلایل گفته شده RNN ناتوانی قابل توجهی نسبت به dataset های بزرگ و دنباله های بزرگ دارند و آموزش خیلی کند اتفاق میفتد.

در مقابل شبکه های Transformer به دلیل مکانیسم توجه امکان موازی سازی دارند و بسیار مقیاس پذیر خواهند بود نسبت به dataset های بزرگ و دنباله های بزرگ. تحقیقات نشان داده است که هر چه قدر مدل های این شبکه بزرگتر باشند کارایی بیشتری خواهند داشت.

یک نگاه دیگر به این ۲ شبکه میتواند اینگونه گفته شود که RNN یک نگاه محلی به دنباله ها دارند یعنی با یک چشم انداز محلی به دنباله ها نگاه میکنند. پردازش هر المان به

صورت ترتیبی خواهد بود و مفهوم جمله را به صورت تدریجی درک میکنند. اما شبکه های Transformer با یک چشم انداز سراسری به دنباله ها نگاه میکنند به سبب استفاده از مکانیسم self-attention. هر المان به اطلاعات سایر المان ها دسترسی دارد و میتوانند ارتباط های پیچیده بین المان ها را بهتر متوجه شوند.

به طور خلاصه

- ✓ شبکه های Transformer به دلیل استفاده از مکانیسم self-attention اجازه میدهند هر المان با سایر المان های دنباله ارتباط برقرار کند و فرقی نمیکند فاصله این ۲ المان چه قدر باشد.
- ✓ با مشکلاتی از قبیل محو شدن گرادیان روبرو نمیشوند و اطلاعاتی را فراموش نمیکنند در ارتباطات طولانی.
- ✓ قابلیت پردازش موازی دارند برعکس پردازش ترتیبی RNNها.

جدولی از مقایسه بین RNN و Transformer:

ویژگی	RNN	Transformer
پردازش	ترتیبی (هر المان در واحد زمان)	موازی (همه المان ها به صورت همزمان)
وابستگی طولانی	ضعیف (محو شدن گرادیان)	عالی (مکانیسم توجه)
سرعت یادگیری	آهسته	سریع
موازی سازی	محدود	بسیار بالا
مقیاس پذیری	محدود	بسیار بالا
حافظه	پایین	بالا (وزن attention)
چهارچوب	نگاه محلی	نگاه سراسری

بخش دوم

بررسی مدیریت RNN

مکانیسم اصلی

شبکه های RNN از hidden state (معمولا با h_t نمایش داده میشود) استفاده میکنند. این hidden state یک برداری است که در هر گام از دنباله بروزرسانی میشود و به نوعی به عنوان حافظه این شبکه ها استفاده میشود. این hidden state بر اساس ۲ چیز بروزرسانی میشود: ۱. ورودی در گام فعلی از دنباله (x_t) ۲. Hidden state قبلی (h_t-1) که فرمول آن به شکل زیر ساخته میشود:

$$h_t = f(W_{xh} * x_t + W_{hh} * h_{t-1} + b_h)$$

که در آن f همان activation function ما است و دو ماتریس وزن برای ورودی ها و connection های بازگشتی از hidden state قبلی داریم بعلاوه بردار bias. نکته مهم در این شبکه ها این است که از وزن های یکسان برای هر کدام از گام های دنباله استفاده میکنند.

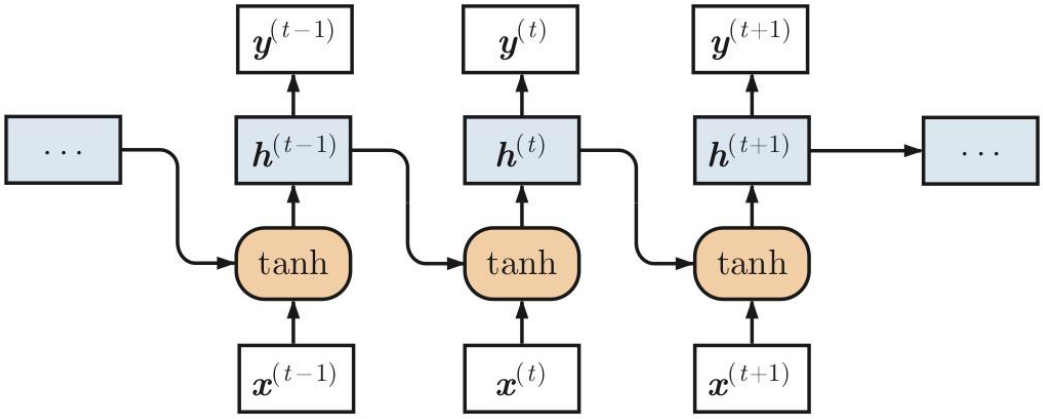


Figure 2: Illustration of the vanilla recurrent neural network.

مشکل شبکه های RNN

چالش اصلی که این شبکه ها با آن روبرو میشوند مسئله BPTT یا Backpropagation Through Time گفته میشود. این BPTT یک الگوریتم برای تنظیم وزن های شبکه های RNN است که در حین یادگیری فراخوانی میشوند. به سبب اینکه در طول گام های زمانی از دنباله اجرا میشود به آن Through time گفته میشود. با استفاده از محاسبات زنجیره ای ما بردار گرادیان تابع loss را

نسبت به وزن ها در هر گام زمانی محاسبه میکنیم. این امر سبب بروز مشکلی به نام محو شدن یا انفجار گرادیان میشود.

محو شدن گرادیان

در محو شدن گرادیان برای وقتی است که مقدار گرادیان کمتر از ۱ شود و حاصل ضرب آن ها عدد بسیار کوچکی میشود وقتی در طی گام های زمانی به عقب حرکت میکند و گرادیان دچار محو شدگی میشود قبل اینکه به بخش های ابتدایی دنباله برسد، به این معنی است که وزن های ابتدایی بروزرسانی نمیشوند و هیچ تاثیری در فرآیند یادگیری نخواهند داشت و به نوعی شبکه دچار فراموش میشود. مشکل دیگر انفجار گرادیان است.

انفجار گرادیان

این حالت وقتی اتفاق میفتد که مقدار گرادیان بیشتر از ۱ باشد، حاصل ضرب آنها عددی بسیار بزرگ میشود در طی گام های زمانی. این باعث میشود که بروزرسانی وزن ها بسیار بزرگ باشد و فرآیند یادگیری ناپایدار میشود و احتمال دارد مدل واگرا شود.

گلوگاه (bottleneck)

یک مشکل دیگری که این شبکه ها دارند این است که hidden state ها از لحاظ ابعاد ثابت هستند یعنی طول بردار آنها ثابت است. این امر باعث کمبود حافظه میشود زیرا هر چه قدر دنباله بزرگتر شود RNN نیاز دارند که اطلاعات بیشتری را در این بردار با طول ثابت جا بدهند، در حین این فرآیند مقداری از اطلاعات از بین میرود.

وجود bias بالا نسبت به ورودی های جدیدتر

به سبب روشی که hidden state بروزرسانی میشوند این شبکه ها تاثیر بیشتری از ورودی های جدید که تازه وارد شده اند میپذیرند نسبت به المان های اول دنباله.

نحوه مدیریت وابستگی‌های طولانی مدت

این hidden state ها سعی میکنند اطلاعات موجود در دنباله را از ابتدا تا لحظه فعلی از دنباله را encode کنند. در تئوری اگر hidden state ها بتوانند همه اطلاعات قابل استفاده را در خودشان نگه دارند میتوانند وابستگی های طولانی مدت را بین المان های دنباله بدست بیاورند حتی اگر اون ۲ المان با همدیگر فاصله داشته باشند.

بررسی مدیریت شبکه های Transformer

مکانیسم self-attention

این مکانیسم اصلی ترین نوآوری در این شبکه ها حساب میشوند. فرق اساسی این شبکه ها با RNN در این است که دیگر ما پردازش ترتیبی دنباله ها را نداریم. کل دنباله به صورت موازی پردازش میشوند.

امتیاز attention

برای هر المان از یک دنباله این مکانیسم یک attention score را محاسبه میکند که بیان گر میزان ارتباط سایر المان ها به المان فعلی خواهد بود.

محاسبه Queries, Keys and Values

هر المان به این ۳ بردار تبدیل میشوند این ۳ بردار حاصل ضرب embedding هر المان در ماتریس وزن خواهند بود.

محاسبه attention

امتیاز attention توسط ضرب داخلی بردار query و key خواهد بود فرمول آن به صورت زیر خواهد بود:

$$Attention(Q, K) = softmax(QK^T / sqrt(d_k))$$

بعد از این مرحله خروجی self-attention جمع وزن بردار value همه المان یک دنباله خواهد بود. یعنی اینکه المان های دیگر چه قدر روی المان فعلی تاثیر دارند بر اساس میزان ارتباط آنها وزن اختصاص داده میشود.

مزایا این شبکه‌ها

اصلی ترین مزیت این شبکه همان مکانیسمی است که گفتیم که اجازه میدهد به صورت مستقیم دو المان از دنباله با هم ارتباط داشته باشند و تبادل اطلاعات صورت بگیرد بدون در نظر گرفتن فاصله آنها. و همچنین هیچ گلوگاهی وجود ندارد.

همچنین Transformerها یک نگاه سراسری به کل دنباله دارند یا به اصطلاح global receptive field دارند یعنی هر المان از دنباله میتواند به کل دنباله دید داشته باشد. به خاطر وجود مکانیسم self-attention امکان موازی سازی وجود دارد و آموزش سریعتر میشود.

این شبکه ها دیگر از مشکلاتی همچون محو شدگی گرادیان یا انفجار گرادیان رنج نمیبرند به سبب اینکه گرادیان دیگر لازم نیست از طریق زنجیره ای از محاسبات recurrent به عقب برگردد. همچنین با استفاده از positional encoding ترتیب درست دنباله را حفظ میکنند.

وجود قابلیت multi-head attention اجازه میدهد مدل گونه های مختلفی از ارتباطات را به صورت همزمان بدست بیاورد به عنوان مثال اگر ما یک جمله داشته باشیم یک از اینها میتواند فقط روی ارتباطات nounها کار کند یکی روی ارتباطات verb با adverb و به صورت همزمان شروع به استخراج ارتباطات کنند. یعنی بعضی از head ها میتوانند ارتباطات محلی را بدست بیاورند در حالی که بقیه headها دارند وابستگی طولانی مدت را بدست میاورند.

چالش‌ها

یکی از معایب این شبکه‌ها وجود مرتبه نمایی برای محاسبات self-attention است به طور که مرتبه زمانی این شبکه ها از مرتبه n^2 است. همچنین این شبکه نیاز به حافظه بیشتری دارند.