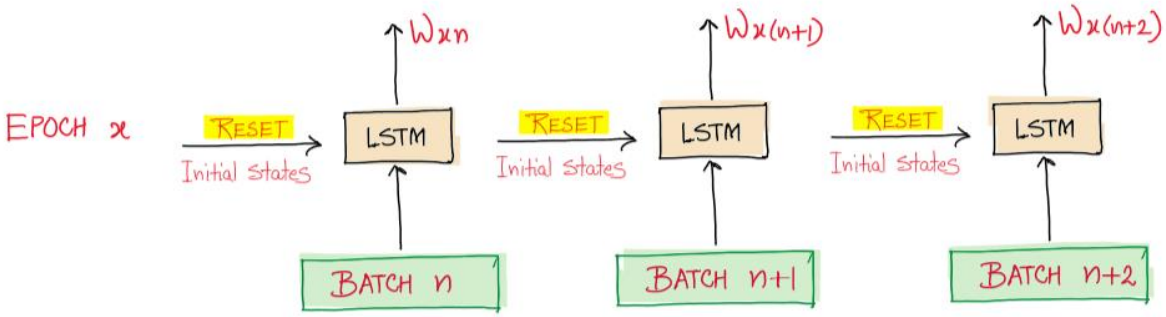
Exercise 1

What's the difference between Stateful RNN vs. Stateless RNN? What are their pros and cons?

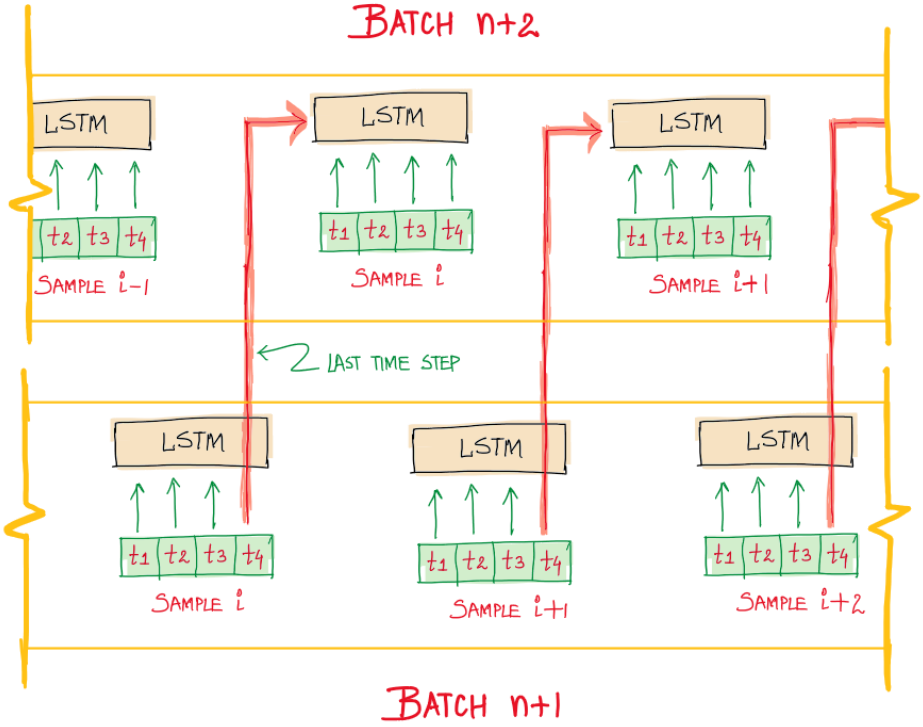
برای یافتن تفاوت این دو ابتدا نیاز است به معماری انها بپردازیم. الگوریتم‌های مدل‌سازی دنباله‌ای، بسته به معماری مورد استفاده در حین آموزش شان دو دو دسته ی Stateful و Stateless قرار میگیرند. این در حالی است که بچ هایی برای آموزش ایجاد میشود، هیچ رابطه متقابلی در بین دسته ها وجود ندارد و هر دسته مستقل از یکدیگر است.

فرایند اموزش معماری Stateless RNN بدین صو رت است:



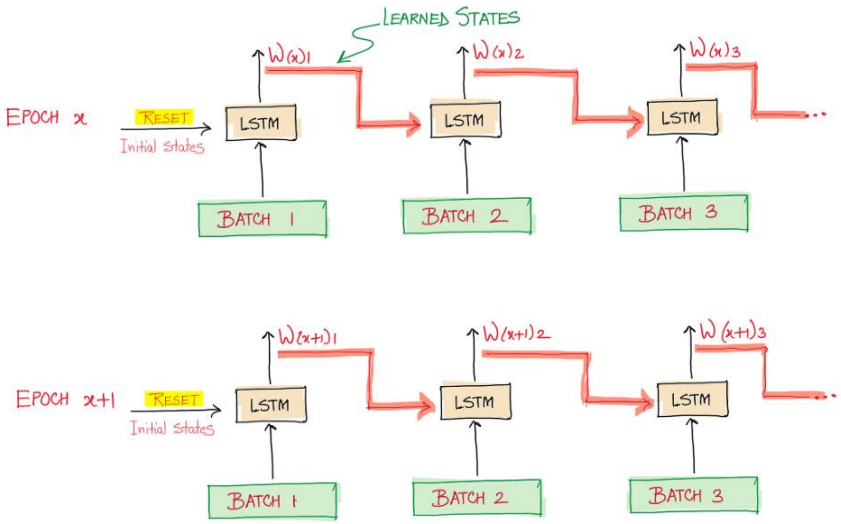
همان طور که در نمودار بالا دیده میشود ، هر بار که بچ جدیدی پردازش می‌شود، حالت‌های اولیه به صفر ریست میشوند، بنابراین در این مدل از حالت‌های از قبل آموخته‌شده استفاده نمی‌شود. این کار مدل را مجبور می کند تا آموخته های دسته های قبلی را فراموش کند.

فرایند اموزش معماری Stateless RNN بدین صو رت است:



همان طور که دیده میشود، سلول‌ها و حالت‌های پنهان این مدل برای هر دسته با استفاده از حالت‌های آموخته‌شده از دسته قبلی، مقداردهی اولیه می‌شوند، در نتیجه مدل وابستگی را در بین دسته‌ها یاد می‌گیرد. با این حال، در آغاز هر دوره، حالت‌ها ریست می‌شوند.

اگر نگاه دقیق تری به مدل بیندازیم به چنین نموداری میرسیم:



در اینجا وضعیت نمونه واقع در ایندکس آی،X[i] در محاسبه نمونه X[i + bs] در بچ بعدی استفاده خواهد شد. به طور دقیق تر، آخرین حالت برای هر نمونه در ایندکس آی در یک بچ به عنوان حالت اولیه برای نمونه ایندکس آی در بچ بعدی استفاده می شود. در نمودار، طول هر دنباله نمونه 4 (timesteps) است و مقادیر حالت های مدل در گام زمانی t=4 برای مقداردهی اولیه در دسته بعدی استفاده می شود.

پس حال تفاوت این دو را در میابیم. جوری که این دو معماری با هم تفاوت دارند، جوری است که در آن حالت‌های مدل (مرتبط با هر بچ) با پیشرفت ترین کردن از یک دسته به دسته دیگر، مقداردهی اولیه می‌شوند(که البته این نباید با پارامترها/وزن‌ها اشتباه گرفته شود، که به هر حال در کل فرآیند تمرین منتشر می‌شوند و این هدف مدل از ترین کردن است).

پس در مدل Stateless RNN در پایان هر بچ، "وضعیت شبکه ریست شده است" این در حالی است که برای مدلStateful RNN وضعیت شبکه برای هر بچ حفظ می شود و سپس باید در پایان هر دوره به صورت دستی ریست شود.

در مدل Stateless RNN ، مدل پارامترها را در بچ اولی به روز می کند و سپس، حالت های پنهان و حالات سلول (معمولاً همه صفرها) را برای بچ دومی آغاز می کند، در حالی که در مدل هایStateful RNN از آخرین حالت های مخفی خروجی بچ اولی و حالت های سلولی به عنوان حالت های اولیه برای بچ دومی استفاده می کند.

یک Stateless RNN از شما می‌خواهد که داده‌های خود را به شیوه‌ای خاص ساختار دهیدو به نوبه خود عملکرد بسیار بیشتری دارد، در حالی که یک Stateful RNN می‌تواند مراحل زمانی متفاوتی داشته باشد، اما با جریمه عملکرد بالا.

اگر به مزایا و معایب این دو مدل بپردازیم به اینجا میرسیم که به طور کلی از مزیت های مدل Stateless RNN این است که عملکرد بیشتری دارد. این در حالیست که مدلStateful RNN را باید به صورت دستی یک دوره در یک زمان آموزش دهید و حالت داخلی را بعد از هر دوره با ()reset_states ریست کنید. از مزایای مدل Stateful RNN نسبت به میتوان به این اشاره کرد که این مدل ها معمولا اندازه شبکه کوچک تر و در نتیجه مدل زمان ترین کمتری دارند. از معایب مدلStateless RNN این است که ترین کردن این مدل ها بیشتر طول میکشد و به حجم مموری بیشتری نیاز داریم، به سختی میتوان در ان ها از تکنیک دارپ اوت استفاده کرد و این مدل ها به شدت به وزن های رندومی که مقداردهی اولیه میشوند حساس هستند.

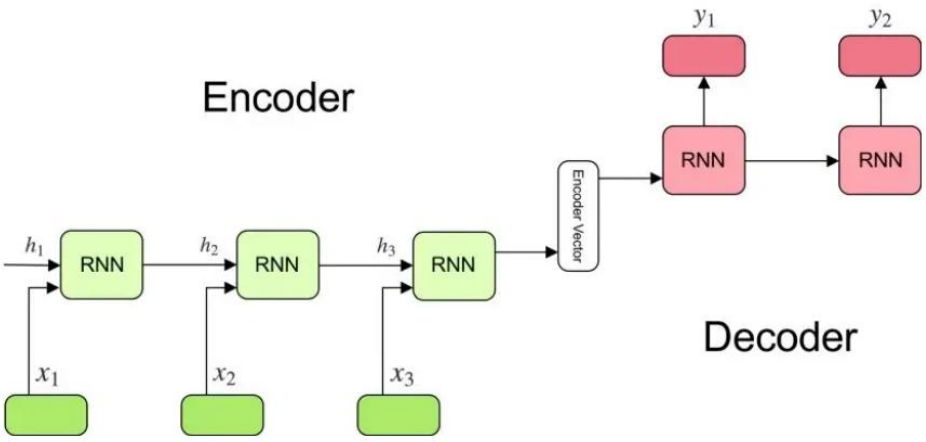
به طور کلی هنگامی که دو دنباله در دو بچ دارای اتصال هستند (مثلاً قیمت یک سهام)، بهتر است ازStateful RNN استفاده کنید، در غیر این صورت (مثلاً یک دنباله یک جمله کامل را نشان ی دهد) باید از Stateless RNN استفاده کنید. اکثر مردم در عمل از Stateless RNN استفاده می کنند، زیرا اگر از Stateful RNN استفاده کنیم، در مرحله تولید، شبکه مجبور می شود با دنباله های طولانی بی نهایت سر و کار داشته باشد، و ممکن است رسیدگی به این موضوع دشوار باشد.

Exercise 2

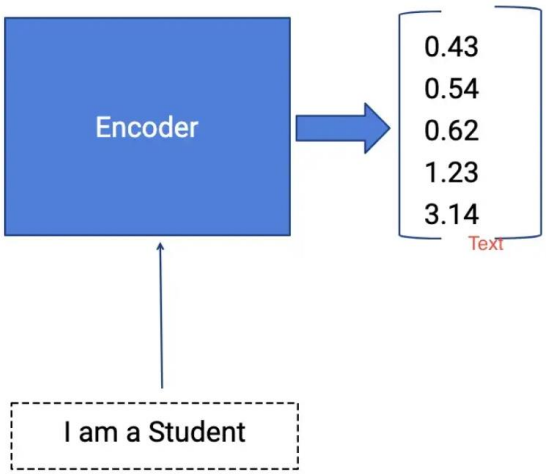
Explain the main difference between Encoder-Decoder RNNs such as Seq2Seq model vs plain RNNs.

ابتدا به بررسی مدل Encoder-Decoder RNN میپردازیم. این مدل شامل دو شبکه عصبی بازگشتی است که به عنوان یک رمزگذار و یک جفت رمزگشا عمل می کنند. رمزگذار یک دنباله منبع با طول متغیر را به یک بردار با طول ثابت نگاشت می کند و رمزگشا بازنمایی برداری را به دنباله هدف با طول متغیر نگاشت می کند. مدل Encoder-Decoder RNN اغلب شامل پیش‌بینی مقدار بعدی در یک دنباله با ارزش واقعی یا خروجی گرفتن از یک لیبل کلاس برای یک دنباله ورودی است. این اغلب به صورت دنباله ای از یک مرحله زمانی ورودی به یک مرحله زمانی خروجی (مثلاً یک به یک) یا چندین گام زمانی ورودی به یک گام زمانی خروجی (چند به یک) تنظیم می شود.

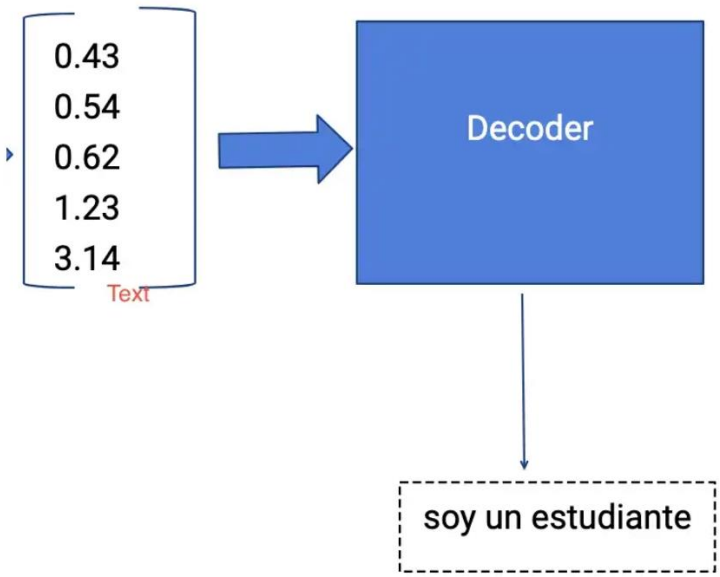
نوع چالش برانگیزتری از مسئله پیش بینی دنباله وجود دارد که یک دنباله را به عنوان ورودی می گیرد و به پیش بینی توالی به عنوان خروجی نیاز دارد. به این مشکلات پیش‌بینی ترتیب به ترتیب یا به اختصار seq2seq گفته میشود. برای درک کامل منطق ساختار مدل، به تصویر زیر میپردازیم.



از کنار هم قرار گرفتن چندین سلول ار ان ان رمزگذار تشکیل میشود. ار ان ان هر ورودی را به ترتیب می خواند. برای هر مرحله زمانی (هر ورودی)، وضعیت پنهان (بردار پنهان) اچ مطابق با ورودی آن مرحله زمان X[i] به روز می شود. پس از خواندن همه ورودی‌ها توسط مدل رمزگذار، حالت پنهان نهایی مدل، متن/خلاصه کل توالی ورودی را نشان می‌دهد. مثالی را در نظر بگیرید. اگر دنباله ورودی " من دانشجو هستم " باشد، برای مدل رمزگذار چهار مرحله زمانی / توکن وجود خواهد داشت. در هر مرحله زمانی حالت پنهان اچ با استفاده از حالت پنهان قبلی و ورودی فعلی به روز می شود.

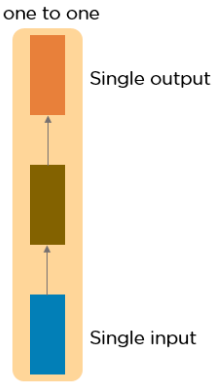


رمزگشا با پیش بینی خروجی بعدی با توجه به حالت پنهان، دنباله خروجی را تولید می کند. ورودی برای رمزگشا، بردار پنهان نهایی است که در انتهای مدل رمزگذار به دست می آید. هر لایه دارای سه ورودی خواهد بود، بردار پنهان از لایه قبلی ht-1 ، لایه قبلی خروجی yt-1 و بردار مخفی اصلی اچ.



حال که توصیف این مدل تمام شد به توصیف مدل بعدی میرسیم. میدانیم مدل ار ان ان چگونه کار میکند. این مدل انواع مختلفی دارد. یک به یک، یک به چند، چند به یک و چند به چند از انوالع ان میباشد.

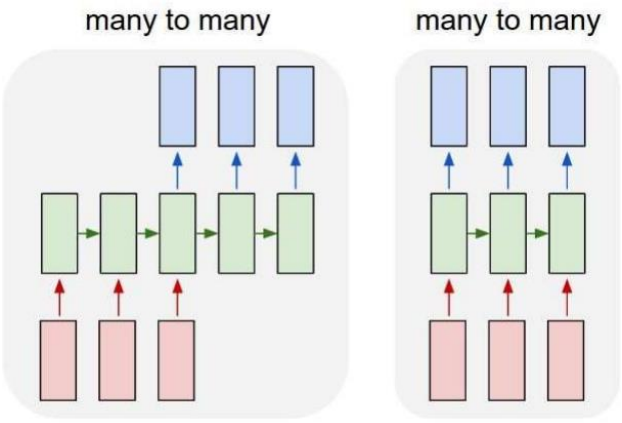
یک به یک : این نوع شبکه عصبی به عنوان شبکه عصبی وانیلی نیز شناخته می شود و برای مسائل یادگیری ماشین که یک ورودی و یک خروجی دارند به کار می رود.



یک به چند : این شبکه عصبی بازگشتی دارای یک ورودی و چند خروجی است. یک نمونه آن، شرح نویسی عکس است.

چند به یک : این نوع مدل دنباله ایی از ورودی ها را می گیرد و یک خروجی تولید می کند. تحلیل احساسات مثال خوبی از این نوع شبکه است که یک جمله را به عنوان ورودی می گیرد و آن را با احساس مثبت یا منفی طبقه بندی می کند.

چند به چند : دنباله ایی از ورودی ها را می گیرد و دنباله ایی از خروجی ها را تولید می کند. ترجمه ماشینی نمونه ایی از این نوع شبکه است. درکل دومدل از این معماری موجود است، که در زیر آمده است.



حال که با تفاوت کلی در ساختار این دو مدل اشنا شدیم به کاربرد شان میپردازیم. به طور کلی زمانی ترجیح میدهیم از این مدل استفاده کنیم که وظیفه ی پیش بینی توالی یا یک سیگنال دوره ای با استفاده از یک ار ان ان را داشته باشیم. ار ان ان نوع اول یک به یک یا همان وانیلی به خوبی کار می کند، اما در هنگام تلاش برای "نگه داشتن در حافظه" وقایع مثلاً بیش از 20 قدم به عقب، کمی مشکل دارد. اگر بخواهیم یک سیستم ترجمه ماشینی داشته باشیم، استفاده از Encoder-Decoder RNN راه بهتری است.

Exercise 3

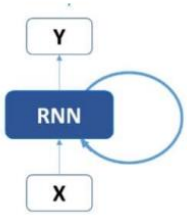
Suppose we want to build a gated RNN cell that sums its inputs over time. What should be the gating values be? To focus on the gating aspect, your design can change the activation function of the RNN cell (e.g., replace tanh by linear).

For the LSTM architecture, what should be the value of the input gate and the forget gate?

For the GRU architecture, what should be the value of the reset gate and the update gate?

میدانیم فرمول مدل ار ان ان به شکل زیر است.

$$h_t = f(h_{t-1}, x_t)$$
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$



اگر این ورودی ها را با گذشت زمان جمع کنیم به فرمول زیر میرسیم:

$h_t = linear(h_{t-1} + x_t)$ $\quad if\ w_{hh} = w_{hx} = w_{hy} = 1 \quad$ $h_t = h_{t-1} + x_t = y_t$

$if\ t = 0: \quad y_0 = x_0$

$if\ t = 1: \quad y_1 = h_0 + x_1 = x_0 + x_1 \quad \Rightarrow \quad y_{(t)} = h_{(t-1)} + x_t$

حال میدانیم مدل ال ای تی ام از اکتیویشن فانکشن زیگموید استفاده میکند. این فانکشن برای جمع ورودی ها فانکشن مناسبی نیست پس تابع اکتیویشن را به دلخواه تغییر میدهیم. اکتیویشن فانکشن صفر را برای یک تابع و برای تابع دیگر اکتیویشن فانکشن 1 را قرار میدهیم.

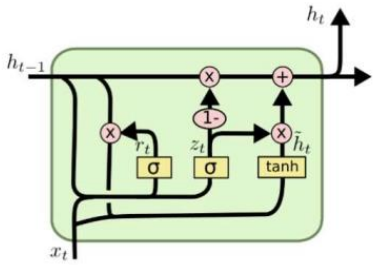
$f_{(t)} = 0$

$g_{(t)} = 1$

$w_c = 1 \ , \quad b_c = 0 \quad => \quad C'_{(t)} = linear(h_{t-1} + x_t) = h_{t-1} + x_t$

$C_t = C'_t = h_{(t-1)} + h_{(t)} \quad =>$ از انجایی که به همان تابع قبلی رسیدیم نشان میدهد مدل درست کار میکند.

اکنون به معماری GRU میپردازیم. برای ساختار این معماری تصویر و فرمول های زیر را داریم.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

اگر همانند قسمت قبل اکتیویشن فانکشن را با اکتیویشن فانکشن یک جایگزین کنیم خواهیم داشت :

$z_{(t)} = 1$

$=>\quad h'_{(t)} = linear(W.[h_{(t-1)}, x_{(t)}])$

$W = 1 \quad => \quad h'_{(t)} = linear\big(h_{(t-1)} + x_{(t)}\big) + x_{(t)} = h_{(t-1)} + x_{(t)} \quad , \quad h_{(t)} = h'_{(t)} = h_{(t-1)} + x_{(t)}$

از انجایی که مدل دوباره فرمول را بدست اورد و درست حساب کرد پس مدل درست خواهد بود.