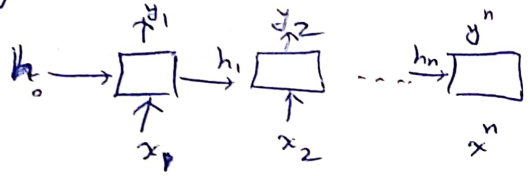
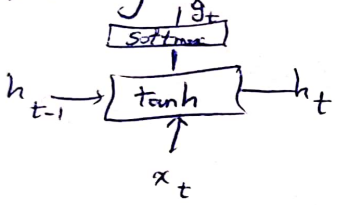
(i) شبکه RNN که پدر GRU و LSTM است شامل تعدادی نورون هستی است که این نورون ها ی مختلف تکراری شوند به طوری که انگار در طول زمان تکراری شوند شکل این شبکه را بصورت ساده تر رسم می کنند



سپس داریم به زبان ریاضی:

$$h_t = g_1(w_a[h_{t-1}, x_t] + b_a)$$
$$y_t = g_2(w_y a_t + b_y)$$

حال مشکل RNN ها این است که اگر جمله ما ادامه پیدا کند و بزرگ باشد شبکه نمی تواند اطلاعات را بصورت long_term نگه دارد و مشکل ناپدید شدن گرادیان پیش می آید و واحد RNN به شکل زیاست:



حال برای حل این شکل واحد GRU استفاده می شود که حافظه بهتری دارد.

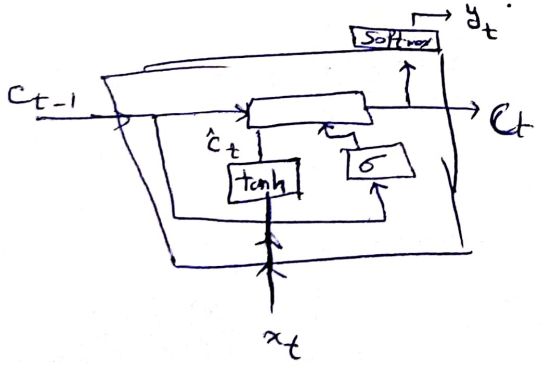
حال برای پیشنهاد دادن state بعدی ورودی جاری، state قبلی را در هم Concat می کنیم و در وزن W ضرب می کنیم

(به اسلایت های جدید c می گوییم که مخفف cell است)

و یک سری گیت اضافه شده برای پیش بردن پذیرش وضعیت بعدی

این معادی برای این که شخص کند از دو بردار استفاده می کند

$$\hat{c}_t = \tanh(w_c[c_{t-1}, x_t] + b_c)$$
$$\Gamma_u = \sigma(w_2[c_{t-1}, x_t] + b_u)$$
$$c_t = \Gamma_u * \hat{c}_t + (1 - \Gamma_u) c_{t-1}$$
$$y_t = \text{softmax}(w_3 c_t + b_o)$$



۱) شبکه LSTM یک نوع شبکه بازگشتی بلندمدت هستند که مشکل Vanishing Gradient را حل کرده. درحقیقت این شبکه می‌توانند اطلاعات دور را به خوبی یاد بگیرند.

۲) تفاوت اصلی GRU و LSTM این است که GRU دو گیت update و reset دارد ولی LSTM سه گیت ورودی و خروجی و فراموشی دارد

3) در LSTM گیت ۲ را داریم که مشخص می‌کند چه مقدار از حافظه گذشته در state فراموش شود و اگر صفر باشد به طور کلی از حافظه از بین خواهد رفت.

4) اگر مجموعه داده‌گان کوچک باشد GRU بهتر عمل خواهد کرد و LSTM با داده‌های زیاد بهتر کار می‌کند

5) GRU کل حافظه را در معرض نمایش می‌گذارد ولی LSTM این کار را انجام نمی‌دهد. و وضعیت‌های پنهان

(2)

استفاده از توابع activation مختلف دیگر مانند Leaky-Relu یا دسته توزیع های ReLU استفاده می کنیم (مانند randomized ReLU / ~~Tabu~~ ReLU)

این توابع به جای Sigmoid و tanh ی تولید شده اند

نرخ یادگیری را کاهش دهیم: اگر نرخ یادگیری خیلی بالا باشد شبکه ممکن است diverge کند و اگر نرخ یادگیری پایین باشد با این که شبکه دیرتر آموزش پیدا می کند اما مشکل دیوانه (انفجار / ناپدید شدن) حل می شود.

از مقداردهی اولیه وزن ها استفاده کنیم (weight initialization): مثلاً از روش Xavier استفاده کنیم

البته به خودی خود خوب نیست ولی با ترکیب روش های دیگر مدل robust تری می شود.

از نرمال سازی بچ استفاده کنیم (batch normalization): با این روش حتی اگر از tanh و Sigmoid استفاده کنیم مدل کمتر دچار مشکل می شود

ساختار شبکه یا Architecture را تعویض کنیم: مثلاً به جای معماری های قدیمی تر از معماری های جدیدتر استفاده کنیم