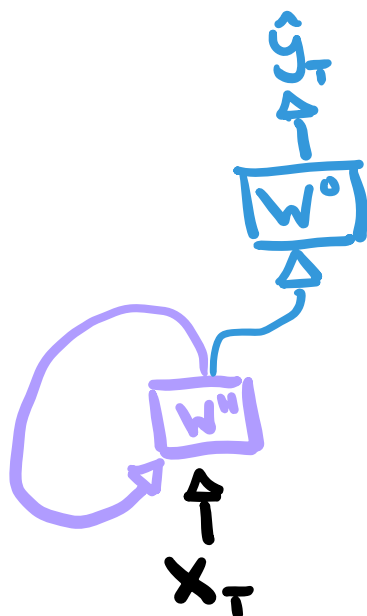
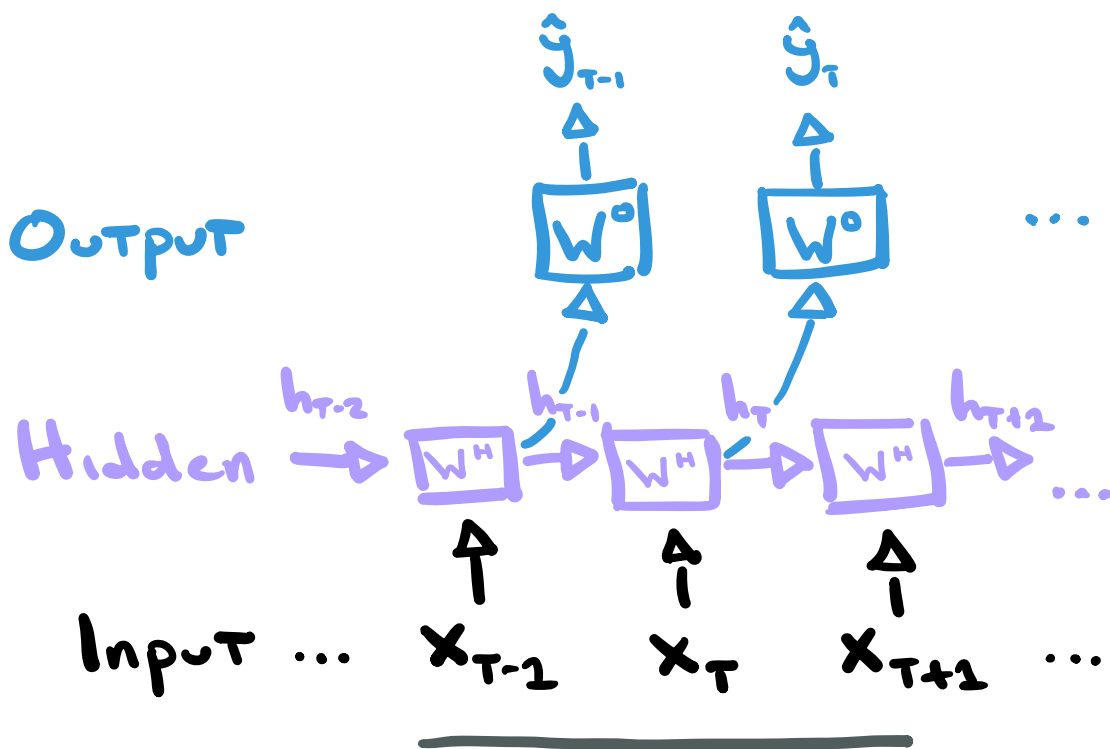


DS 4440

Recurrent Neural Networks 2

Last Time: RNNs for Sequences

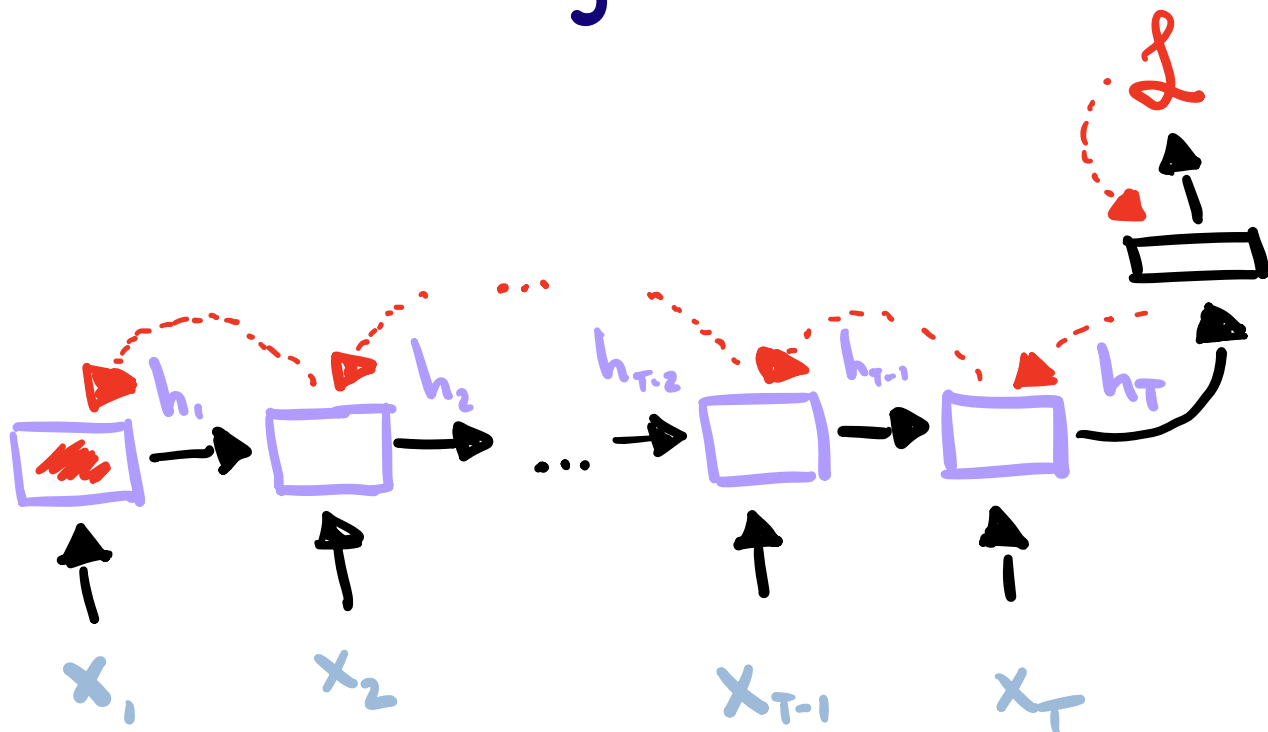


Problem long term dependencies

(0 0 0 0 0)	0
(1 0 0 0 0)	1
...	
(1 0 0 0 0)	1

x y

Signal must flow
Through entire
Sequence!

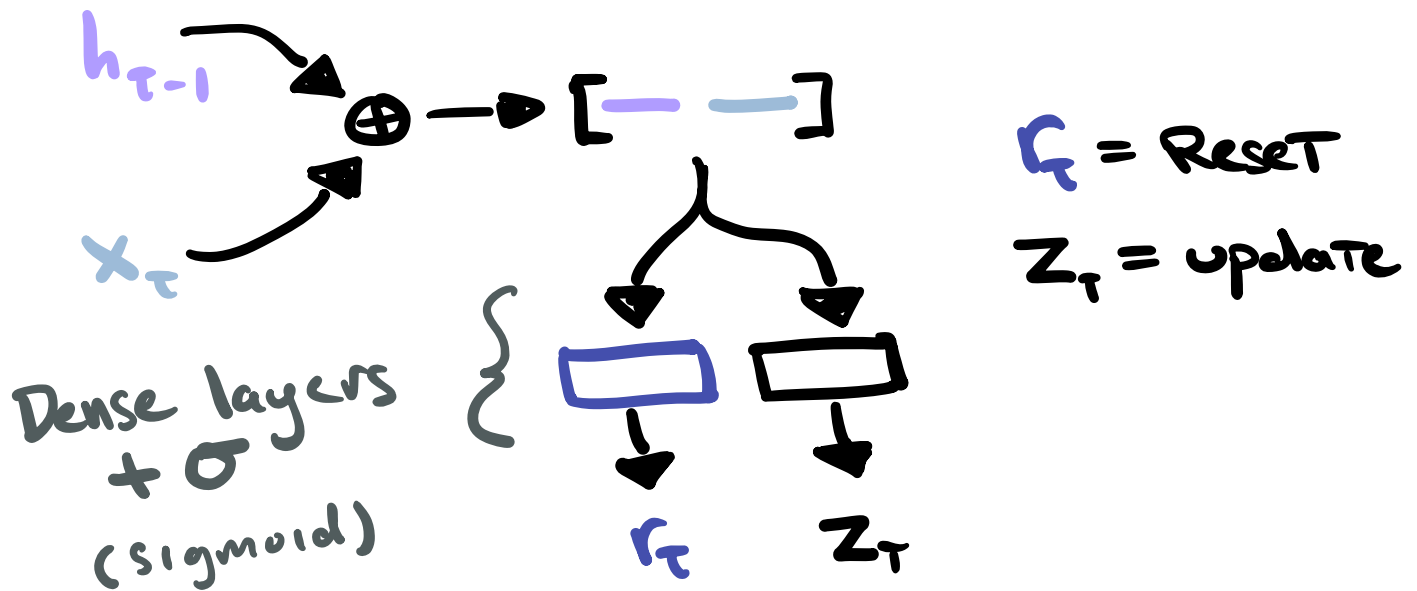


Gated Recurrent Units (GRUs)

Similar to **LSTMs** (next) but
Simpler.

IDEA Introduce gating mechanisms

To allow Model to update h_t
or opt not to (Skip)



$$r_t = \sigma(x_t W_r^x + h_{t-1} W_r^h + b_r)$$

$$z_t = \sigma(x_t W_z^x + h_{t-1} W_z^h + b_z)$$

Reset

$$\tilde{h}_t \leftarrow \tanh(x_t W_h^x + \underbrace{(r_t \odot h_{t-1})}_{\text{Permits forgetting}} W_h^h + b_h)$$

If $r_t \approx 0 \rightarrow$ "resets" h_t

Update

$$h_T \leftarrow \underbrace{z_T \odot h_{T-1}}_{\text{Copy}} + \underbrace{(1-z_T) \odot \tilde{h}_T}_{\text{update}}$$

If $z_T = 1$ Then $h_T = h_{T-1}$.

Let's think about the **grad** here.

$$\frac{\partial h_T}{\partial h_{T-1}} = \underbrace{\text{diag}(z_T)}_{\text{Copy path}} + \underbrace{\text{diag}(h_{T-1} - \tilde{h}_T)}_{h_{T-1} \text{ affects gate}} \frac{\partial z_T}{\partial h_{T-1}}$$

$$+ (1-z_T) \underbrace{\frac{\partial \tilde{h}_T}{\partial h_{T-1}}}_{\text{Through } \tilde{h}_T}$$

Near identity, so **grad** can flow back easily.

Return to "remembering a bit"

(0 0 0 0 0) 0

(1 0 0 0 0) 1

Label is first symbol, x_1 .

So x_1 should be written into h

and held onto.

Consider a vanilla RNN first.

$$h_t = \tanh(x_t W_x + h_{t-1} W_h)$$

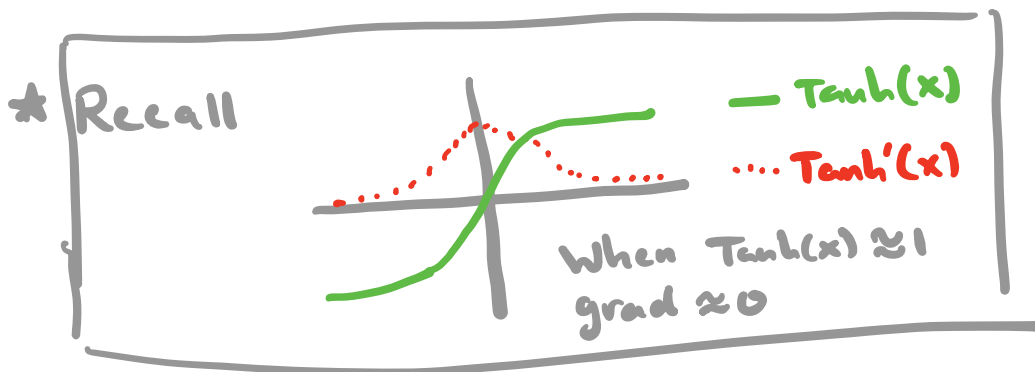
Let $W_x = 3$ (all scalars!)

Then try $W_h = 1$.

$$h: .995 \rightarrow .76 \rightarrow .64 \rightarrow .57 \rightarrow .51$$

by $t=50$, $h = .17$ and $\frac{\partial h_{50}}{\partial h_1} = 4.6 \cdot 10^{-3}$

Try $W_h = 5$. Tanh Saturates*



$$\frac{\partial h_{50}}{\partial h_1} \approx 10^{-149} !$$

So we can remember the bit but
can't learn.

How about the GRU here?

$$W_2^x = -10 \quad W_2^h = 10 \quad b_2 = 0$$

update gate

$$\underbrace{b_r = -10}_{\text{closed reset}} \quad \underbrace{w_h^x = 3}_{\text{candidate}}$$

$$z_\tau = \sigma(-10x_\tau + 10h_{\tau-1})$$

A 1 drives z to 0 $\rightarrow$ Write

if x_τ is 0 $\rightarrow$ Copy

$$r_\tau = \sigma(-10) \approx 0.$$

$$\tilde{h}_\tau = \tanh(3x_\tau) \approx \begin{cases} 1 & \text{when } x_\tau = 1 \\ 0 & \text{otherwise} \end{cases}$$

Let's Trace on $[1\ 0\ 0\ 0\ 0]$.

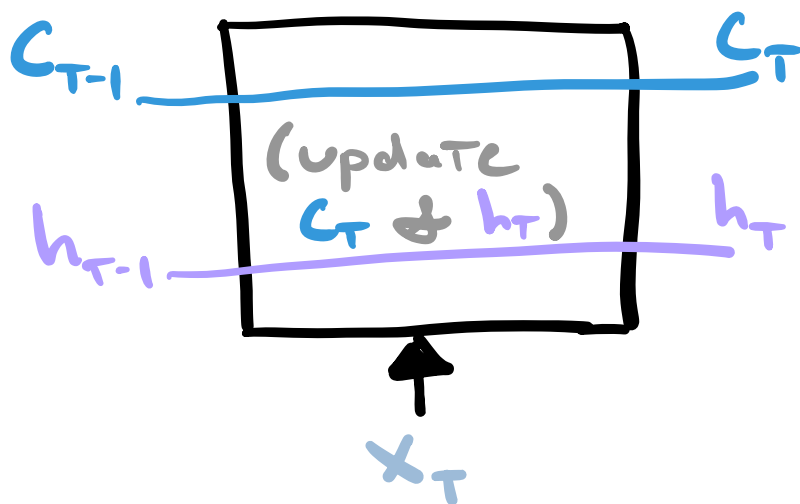
τ	x_τ	z_τ	$\tilde{h}_\tau$	h_τ	
0	-	-	-	0	
1	1	≈ 0	≈ 1	≈ 1	Write!
2	0	≈ 1	0	≈ 1	} Copy
3	0	≈ 1	0	≈ 1	
4	0	≈ 1	0	≈ 1	
5	0	≈ 1	0	≈ 1	

And here, $\frac{\partial h_{50}}{\partial h_1} \approx 1$.

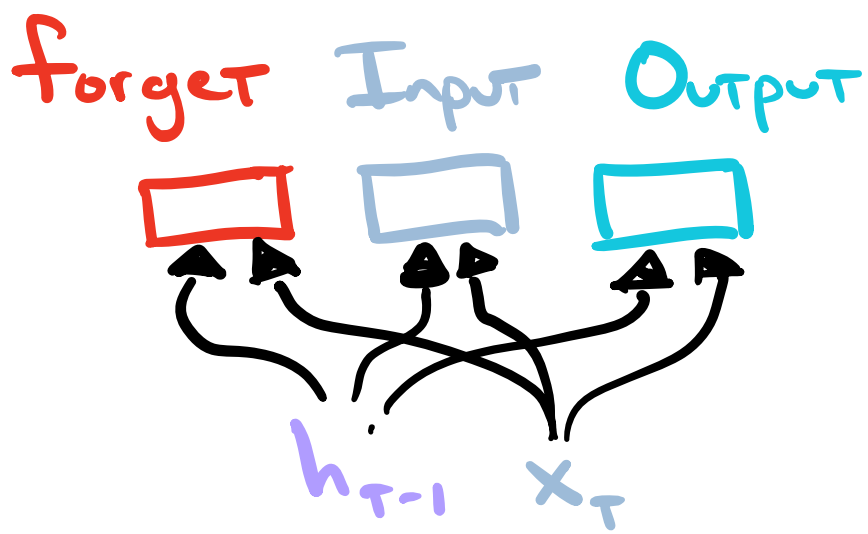
This gives intuition but I've cheated a little - Assuming 1 can only appear at x_1 . To generalize we need more dims, but same idea.

Long Short Term Memory (LSTM)

Like GRUs, use gating for long term dependencies.



Three gates:



Assume batch size $\underline{b}$, $\underline{d}$ hidden dims.

$$x_T \in \mathbb{R}^{b \times d_{in}} \quad h_{T-1} \in \mathbb{R}^{b \times d}$$

$$f_T = \sigma(x_T W_f^x + h_{T-1} W_f^h + b_f)$$

$$i_T = \sigma(x_T W_i^x + h_{T-1} W_i^h + b_i)$$

$$o_T = \sigma(x_T W_o^x + h_{T-1} W_o^h + b_o)$$

$\underbrace{\hspace{1.5cm}}_{b \times d} \quad \underbrace{\hspace{1.5cm}}_{d_{in} \times d} \quad \underbrace{\hspace{1.5cm}}_{d \times d} \quad \underbrace{\hspace{1.5cm}}_{1 \times d}$

Apply These gates to a Candidate
"Memory cell" $\tilde{C}_T$.

$$\tilde{C}_T = \text{Tanh}(x_T W_c^x + h_{T-1} W_c^h + b_c)$$

$\underbrace{\hspace{1.5cm}}$

bxd

Next update The Memory

$$C_T \leftarrow f_T \odot C_{T-1} + i_T \odot \tilde{C}_T$$

Finally, the Output layer, which is a modified version of C_T .

$$h_T \leftarrow o_T \odot \tanh(C_T)$$

Info can be stored in C_T w/o affecting h_T if o_T near 0.

► Why gating makes grad nicer

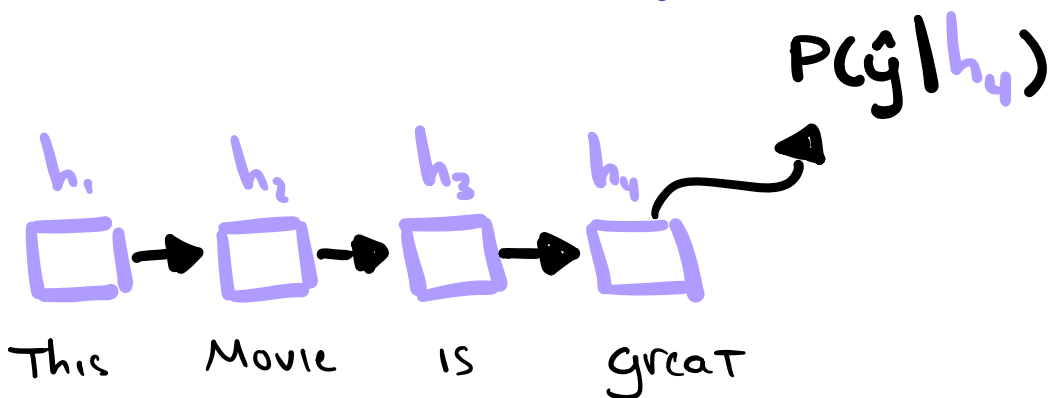
$$\frac{\partial C_T}{\partial C_{T-1}} = \text{diag}(f_T) \quad \text{direct path!}$$

So if f_T entries near 1, prod stays near 1. ("forget gate" probably a bad name; more like a mute?)

RNNs, side by side

	<u>Vanilla</u>	<u>GRU</u>	<u>LSTM</u>
Carried forward	h	h	h, c
gates	None	reset, update	forget, input, output
weight matrices (• Vanilla)	1x	3x	4x
keep vs write	None (always overwrite)	tied (2, 1-2)	separate (f and i)

(Vanilla) Attention!



Idea Weight hidden States

$$a_j = W_a \cdot h_j$$

$\langle 1 \times h \rangle \langle h \rangle$

$$\bar{h} = \sum_{j=1}^n \alpha_j h_j$$

$$\alpha = \text{SoftMax}(a) \quad P(y|\bar{h})$$

More next time ...