# STOR566: Introduction to Deep Learning

## Lecture 10: LSTM and GRU

Yao Li
UNC Chapel Hill

Sep 24, 2024

Materials are from *Deep Learning (UCLA)*

# Problems of Classical RNN

- Hard to capture long-term dependencies

# Problems of Classical RNN

- Hard to capture long-term dependencies

The cat, which already ate ..., was full

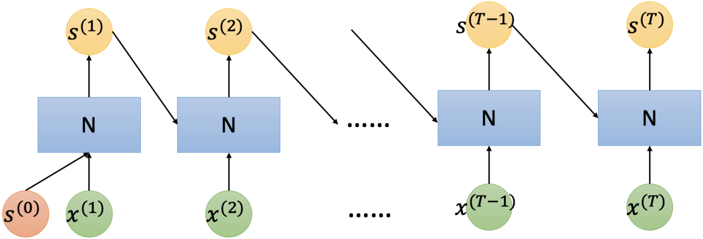
The cats, which already ate ..., were full

# Problems of Classical RNN

- Hard to capture long-term dependencies

The cat, which already ate ..., was full

The cats, which already ate ..., were full



# Gradient Vanishing of RNN

- Hard to solve (vanishing gradient problem)

- $\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \alpha \times \nabla f(\boldsymbol{w}^{t-1})$

# Gradient Vanishing of RNN

- Hard to solve (vanishing gradient problem)

- $\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \alpha \times \nabla f(\boldsymbol{w}^{t-1})$

- $\begin{pmatrix} 0.499999 \\ 2.100001 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 2.1 \end{pmatrix} - 0.01 \times \begin{pmatrix} 0.0001 \\ -0.0001 \end{pmatrix}$

# Gradient Vanishing of RNN

- Hard to solve (vanishing gradient problem)

$$s_t = \sigma(W_1 x_t + W_2 s_{t-1}), \quad o_t = V s_t$$

Let $W = [W_1, W_2]$, the gradient:

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial o_t} \frac{\partial o_t}{\partial s_t} \left( \Pi_{k=2}^{t} \sigma^{'}(W_1 x_k + W_2 s_{k-1}) W_2 \right) \frac{\partial s_1}{\partial W}$$

# Gradient Vanishing of RNN

- Hard to solve (vanishing gradient problem)

$$s_t = \sigma(W_1 x_t + W_2 s_{t-1}), \quad o_t = V s_t$$

Let $W = [W_1, W_2]$, the gradient:

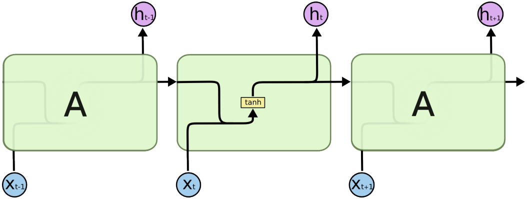
$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$
$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial o_t} \frac{\partial o_t}{\partial s_t} \left( \Pi_{k=2}^{t} \sigma^{'}(W_1 x_k + W_2 s_{k-1}) W_2 \right) \frac{\partial s_1}{\partial W}$$

- Solution:
  - LSTM (Long Short Term Memory networks)
  - GRU (Gated Recurrent Unit)
  - $\cdots$

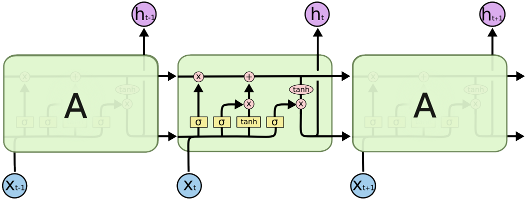
# LSTM

# LSTM

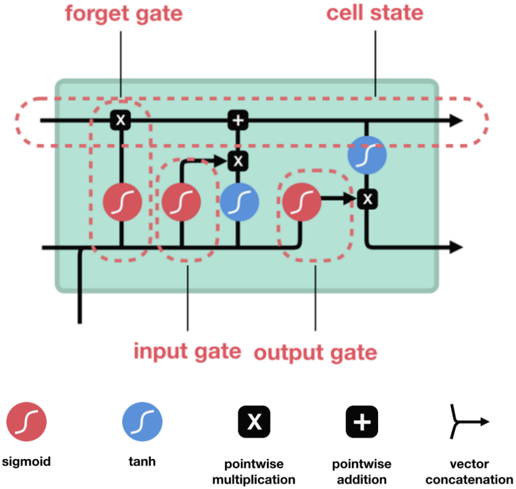
- RNN:



- LSTM:



# LSTM Cell



picture from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

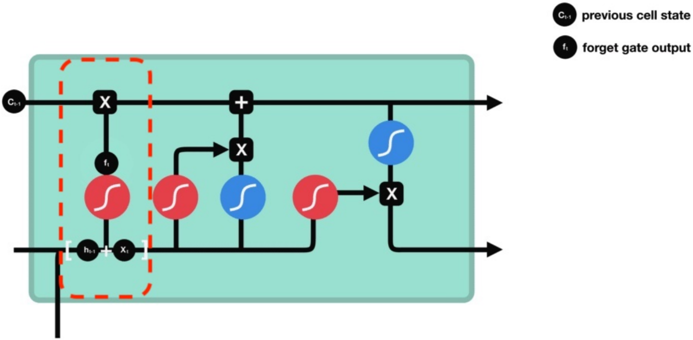
# Cell State and Hidden State

The two hidden states $\boldsymbol{h}^{(t)}$ and $\boldsymbol{c}^{(t)}$ are calculated by:

$$
\begin{aligned}
\boldsymbol{c}^{(t)} &= \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \boldsymbol{z}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{c}^{(t)}),
\end{aligned}
$$

- Cell state: $\boldsymbol{c}^{(t)}$, “memory” of the network
- Hidden state: $\boldsymbol{h}^{(t)}$, information on previous inputs
- $\circ$: point-wise multiplication

# Forget Gate

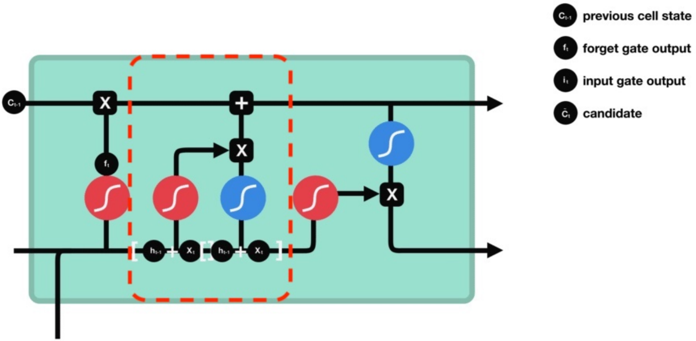


- Compute forget gate output: $\boldsymbol{f}^{(t)} = \sigma_g(\boldsymbol{W}_{1f}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2f}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f)$
- Forget previous information: $\boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)}$
- $\sigma_g$: sigmoid activation

# Cell State

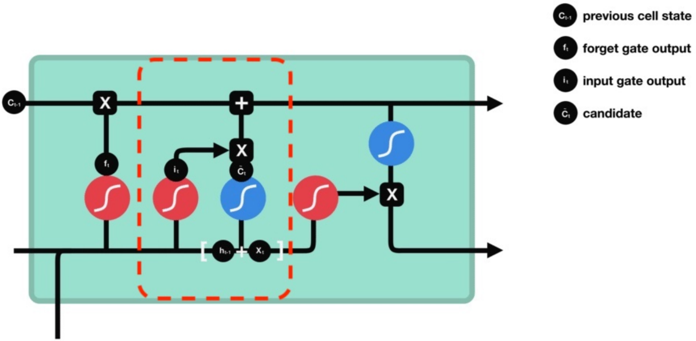
$$\boldsymbol{c}^{(t)} = \underbrace{\boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)}}_{forget \quad gate} + \underbrace{\boldsymbol{i}^{(t)} \circ \boldsymbol{z}^{(t)}}_{input \quad gate},$$

# Input Gate



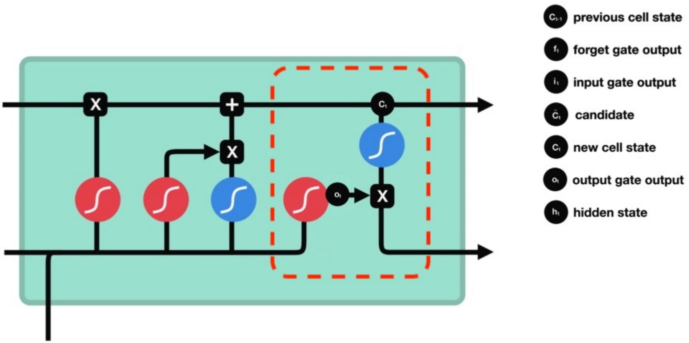
- Determine what to keep: $\boldsymbol{i}^{(t)} = \sigma_g(\boldsymbol{W}_{1i}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2i}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$

# Input Gate



- Determine what to keep: $\boldsymbol{i}^{(t)} = \sigma_g(\boldsymbol{W}_{1i}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2i}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i)$
- Compute tanh output: $\boldsymbol{z}^{(t)} = \tanh(\boldsymbol{W}_{1z}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2z}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_z)$
- $\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \boldsymbol{z}^{(t)}$

# Output Gate



- Decide what to pass into next hidden state: $\boldsymbol{o}^{(t)} = \sigma_g(\boldsymbol{W}_{1o}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2o}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$
- $\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{c}^{(t)})$

# Gradient Vanishing

- Gradient:

$$\frac{\partial \ell^{(T)}}{\partial \boldsymbol{W}} = \frac{\partial \ell^{(T)}}{\partial \boldsymbol{h}^{(T)}} \frac{\partial \boldsymbol{h}^{(T)}}{\partial \boldsymbol{c}^{(T)}} \left( \prod_{j=t+1}^{T} \frac{\partial \boldsymbol{c}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}} \right) \frac{\partial \boldsymbol{c}^{(t)}}{\partial \boldsymbol{W}}$$

# Gradient Vanishing

- Gradient:

$$\frac{\partial \ell^{(T)}}{\partial \boldsymbol{W}} = \frac{\partial \ell^{(T)}}{\partial \boldsymbol{h}^{(T)}} \frac{\partial \boldsymbol{h}^{(T)}}{\partial \boldsymbol{c}^{(T)}} \left( \prod_{j=t+1}^{T} \frac{\partial \boldsymbol{c}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}} \right) \frac{\partial \boldsymbol{c}^{(t)}}{\partial \boldsymbol{W}}$$

- $\boldsymbol{c}^{(j)} = \boldsymbol{f}^{(j)} \circ \boldsymbol{c}^{(j-1)} + \boldsymbol{i}^{(j)} \circ \boldsymbol{z}^{(j)}$

# Gradient Vanishing

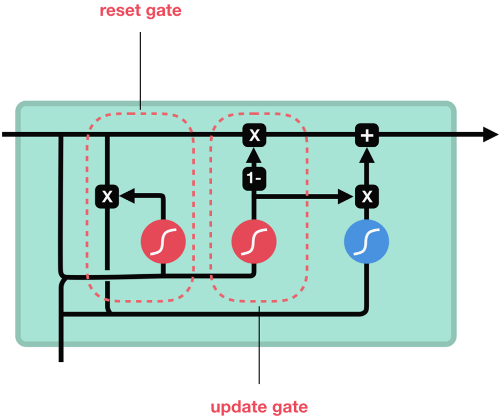
- Gradient:

$$\frac{\partial \ell^{(T)}}{\partial \boldsymbol{W}} = \frac{\partial \ell^{(T)}}{\partial \boldsymbol{h}^{(T)}} \frac{\partial \boldsymbol{h}^{(T)}}{\partial \boldsymbol{c}^{(T)}} \left( \prod_{j=t+1}^{T} \frac{\partial \boldsymbol{c}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}} \right) \frac{\partial \boldsymbol{c}^{(t)}}{\partial \boldsymbol{W}}$$

- $\boldsymbol{c}^{(j)} = \boldsymbol{f}^{(j)} \circ \boldsymbol{c}^{(j-1)} + \boldsymbol{i}^{(j)} \circ \boldsymbol{z}^{(j)}$
- $\frac{\partial \boldsymbol{c}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}} = \boldsymbol{c}^{(j-1)} \times \frac{\partial \boldsymbol{f}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}} + \boldsymbol{f}^{(j)} + \boldsymbol{z}^{(j)} \times \frac{\partial \boldsymbol{i}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}} + \boldsymbol{i}^{(j)} \times \frac{\partial \boldsymbol{z}^{(j)}}{\partial \boldsymbol{c}^{(j-1)}}$
- The summation prevents gradient vanishing.

# Gated Recurrent Unit

# GRU Cell

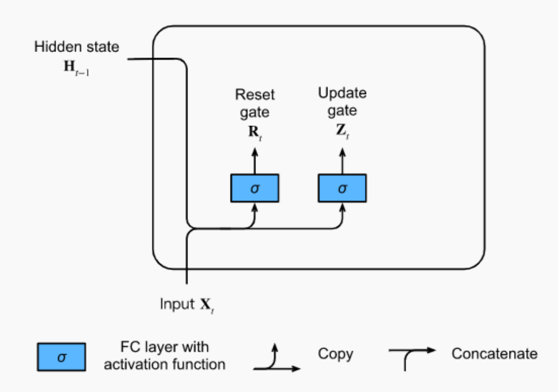


picture from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

# Reset Gate and Update Gate

Reset gate $\boldsymbol{r}^{(t)}$ and Update gate $\boldsymbol{z}^{(t)}$ are calculated by:

$$\boldsymbol{r}^{(t)} = \sigma_g(\boldsymbol{W}_{1r}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2r}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_r),$$
$$\boldsymbol{z}^{(t)} = \sigma_g(\boldsymbol{W}_{1z}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2z}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_z),$$


Hidden state $\mathbf{H}_{t-1}$
Reset gate $\mathbf{R}_t$
Update gate $\mathbf{Z}_t$
σ
σ
Input $\mathbf{X}_t$
σ
FC layer with activation function
Copy
Concatenate

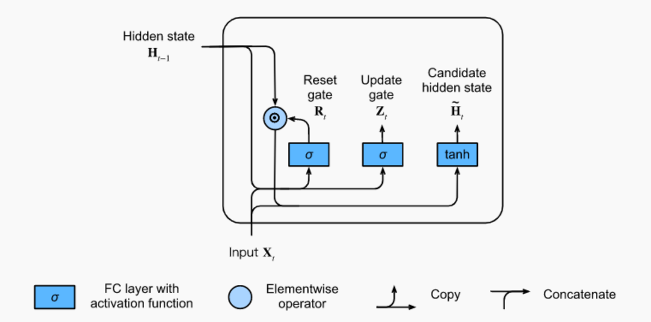
picture from https://d2l.ai/chapter_recurrent-modern/gru.html

# Candidate Hidden State

Candidate hidden state $\tilde{\boldsymbol{h}}^{(t)}$:

$$\tilde{\boldsymbol{h}}^{t} = \tanh(\boldsymbol{W}_{1h}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{2h}(\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{b}_h)$$

- Determine what to be kept from previous hidden state: $\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}$

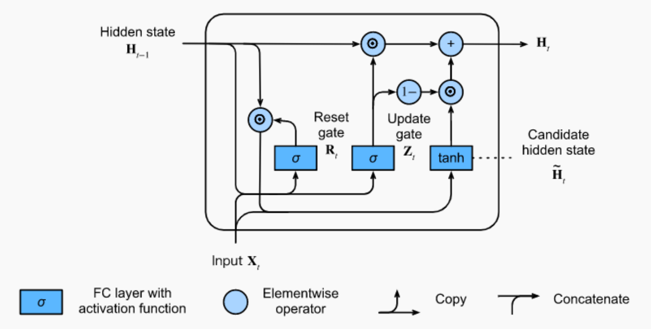

Hidden state $\mathbf{H}_{t-1}$
Reset gate $\mathbf{R}_t$
Update gate $\mathbf{Z}_t$
Candidate hidden state $\tilde{\mathbf{H}}_t$
⊙
σ
σ
tanh
Input $\mathbf{X}_t$
σ
FC layer with activation function
Elementwise operator
Copy
Concatenate

# Final Hidden State

Hidden state $\boldsymbol{h}^{(t)}$:

$$\boldsymbol{h}^{(t)} = \boldsymbol{z}^{(t)} \circ \boldsymbol{h}^{(t-1)} + (1 - \boldsymbol{z}^{(t)}) \circ \tilde{\boldsymbol{h}}^{(t)}$$

- Keep info from previous hidden state: $\boldsymbol{z}^{(t)} \circ \boldsymbol{h}^{(t-1)}$
- Get info from current state: $(1 - \boldsymbol{z}^{(t)}) \circ \tilde{\boldsymbol{h}}^{(t)}$



# Conclusions

- Gradient Vanishing
- LSTM
- GRU

Questions?