[Lec 07]

Vanishing Gradient

→ Think of 'Chain Rule' (Multiplied by smaller values → .. → ...)

→ gets exponentially small as it goes to the shallower layers

- `Pascanu et al`

→ Largest eigenvalue of W_h is < 1 → gradient will shrink exponentially

→ Largest eigenvalue > 1 → exploding gradients

cf. bound 1 (sigmoid nonlinearity)

- **Why it is a problem**

1. Gradient signal from faraway is lost because it is much smaller than the gradient signal from close-by

   → Updated with respect to near effects

2. Gradient can be viewed as a measure of the effect of the past on the future

   a. no dependency between step t and t+n

   b. wrong parameters to capture the true dependency between t and t+n

- RNN better at learning

cf. Syntactic recency(`The writer` of the books is) vs **Sequential recency (The writer of the `books are`)**

Exploding Gradient

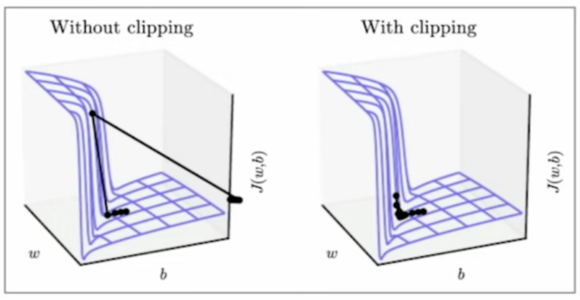
- **Why it is a problem**

→ SGD update becomes too big (lead to bad updates - result in Inf and NaN)

- **Solution) Gradient Clipping**

If the norm of the gradient is greater than some threshold

→ scale it down between applying SGD update



- **Skip Connections (ResNet)**

→ Vanishing/Exploding Gradient is not just the problem for RNNs

→ Can be seen in CNNs and FF Networks (Deep Layers)

Sol) Residual Connections (ResNet) → Identity Connection preserves information v

LSTM (Long Term Short Memory)

→ **RNN with separate memory (motivating idea)**

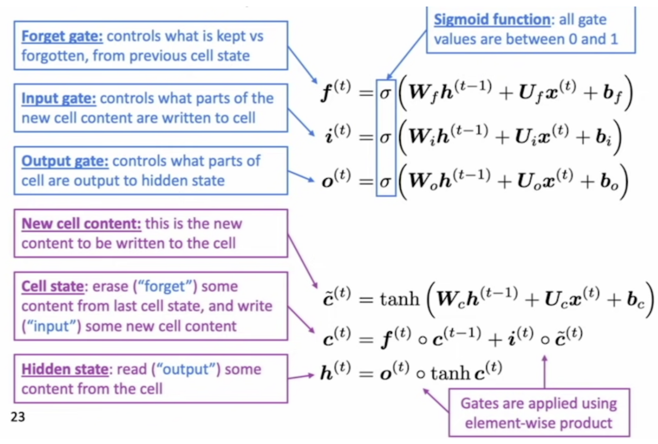
→ Solution for vanishing gradients problem

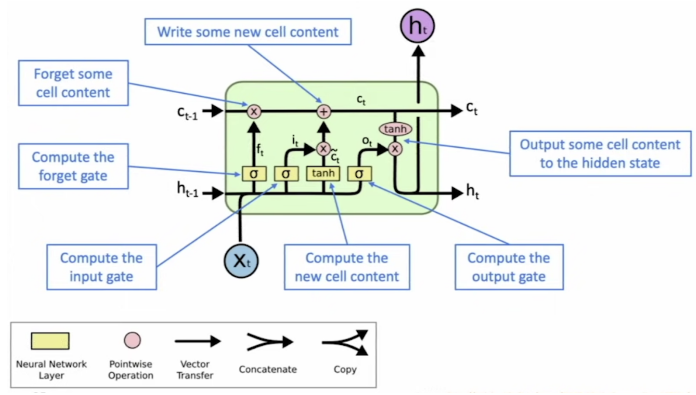
- hidden state h_t and cell state c_t
- LSTM can erase, write and read information from the cell

→ selection of which information is erased/written/read is controlled by three corresponding gates

- open. closed. in-between
- dynamic gates → value computed by current context

Cell State





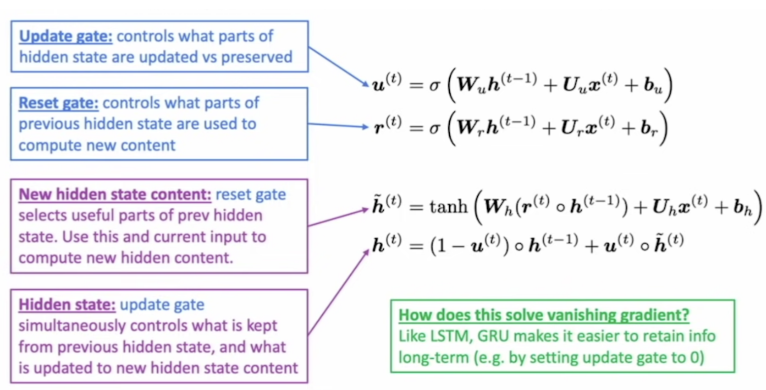
→ LSTM solves Gradient Descent by its `architecture` to preserve information over many timestamps

→ doesn't guarantee entirely

GRU (Gated Recurrent Units)

→ more simple than RNNs

→ No cell state!



LSTM vs GRU

→ Most Widely Used

→ `GRU` : Quicker to compute -fewer parameters

→ No conclusive evidence of outperformance between two.

- **LSTM → Good Default Choice (Long Dependencies & More training Data)**
- **Change to GRU if need more efficiency**

Fancy RNN

- **Bidirectional RNNs**
- **Multi-Layer RNNs**