Recurrent Neural Nets Training

Thang Vu

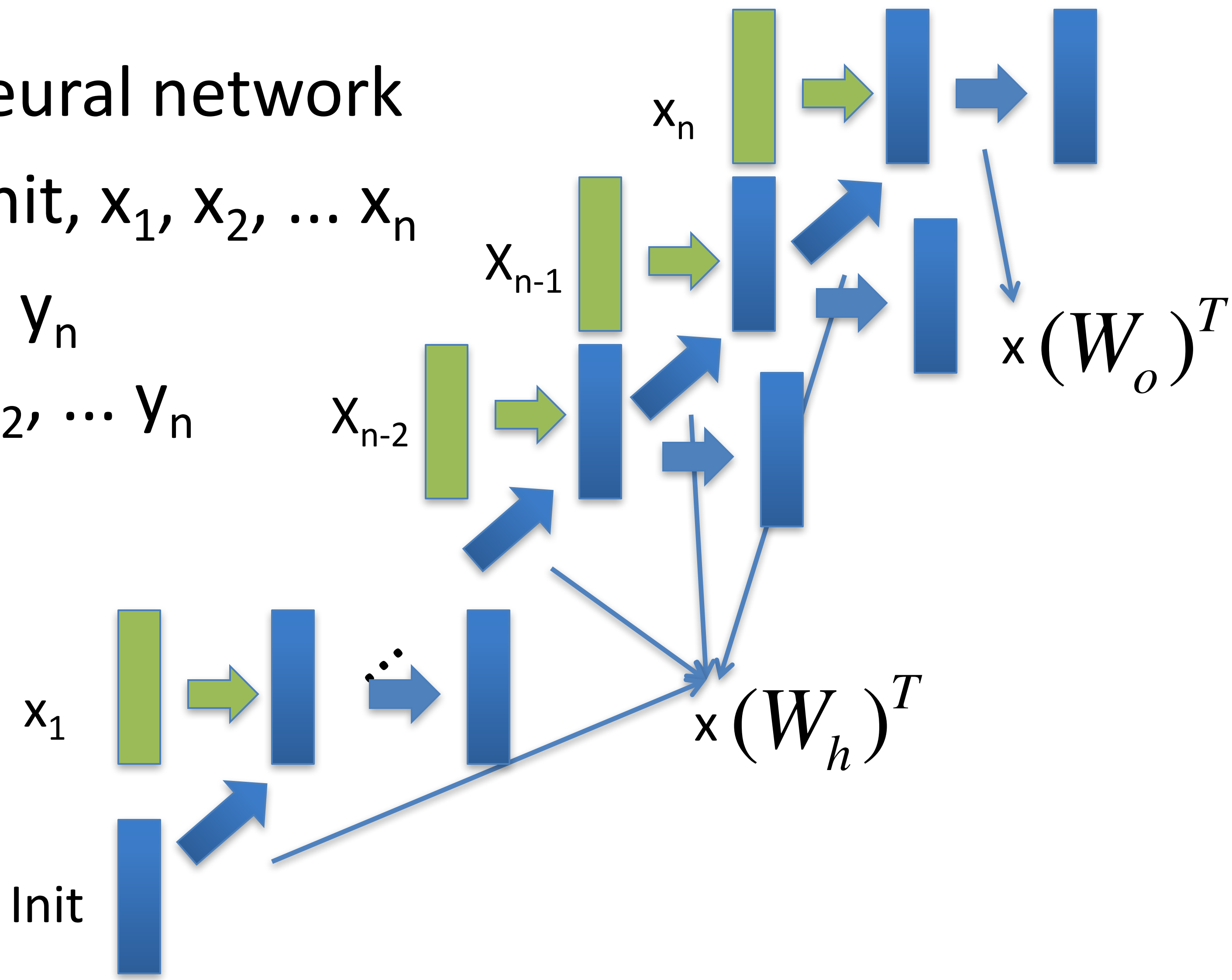
RNN Training

- Cost function is the same as MLP or CNN
  - E.g. cross entropy
- Parameters are W_i, W_o, W_h
- Training algorithm:

  Backpropagation Through Time (BPTT)

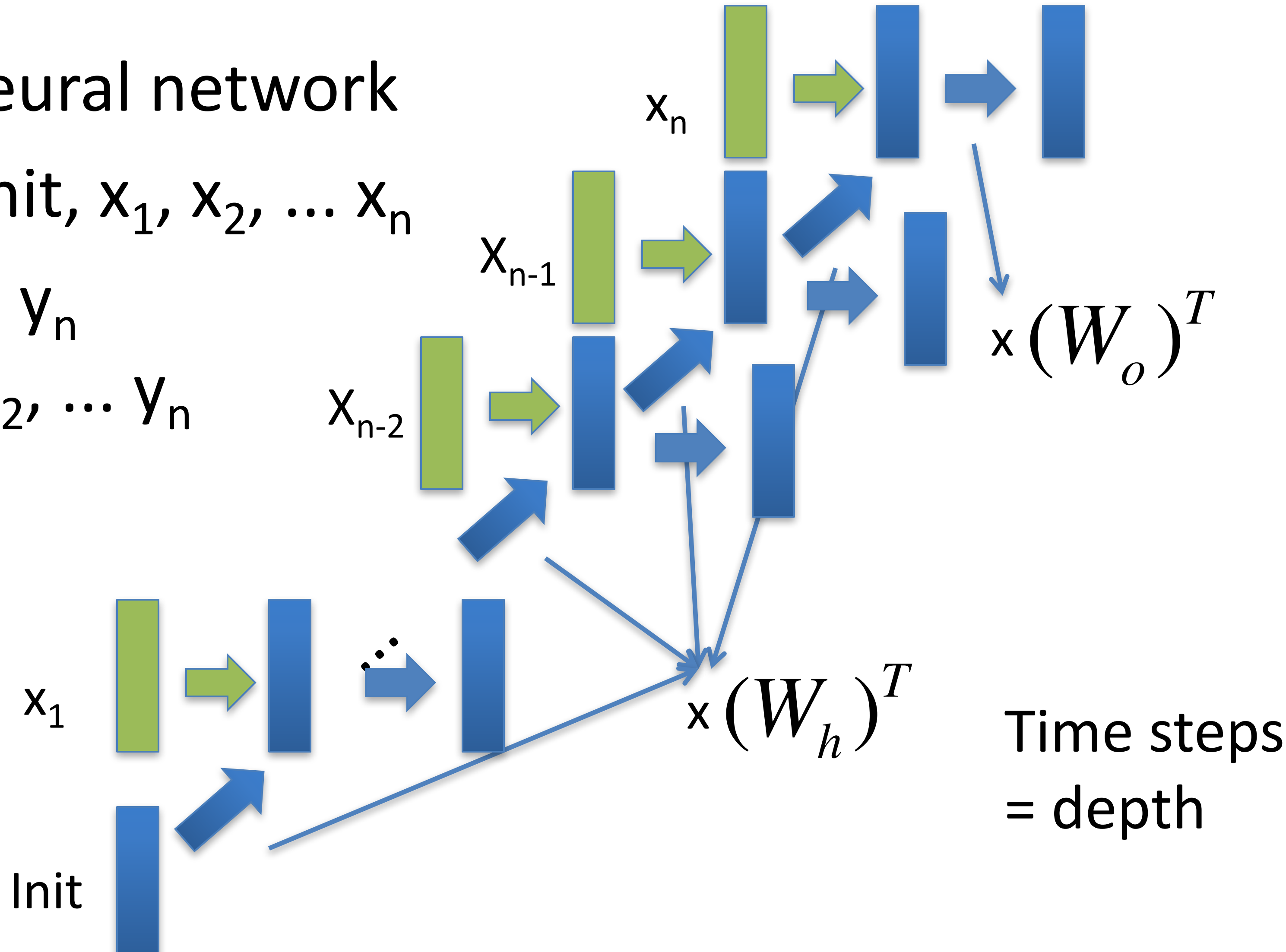
BPTT

- Deep neural network
- Input: Init, x_1, x_2, … x_n
- Output: y_n
  OR y_1, y_2, … y_n



BPTT

- Deep neural network
- Input: Init, x_1, x_2, … x_n
- Output: y_n
  OR y_1, y_2, … y_n



BPTT

- Deep neural network
  - Weights are shared
  - You can define how many time steps you want to look back to compute the gradients
- If $w_1 = w_2$ then $\Delta w_1 = \Delta w_2$

Compute $\frac{\partial C}{\partial w_1}$ and $\frac{\partial C}{\partial w_2}$

Use $\frac{\partial C}{\partial w_1} + \frac{\partial C}{\partial w_2}$ to update w_1 and w_2

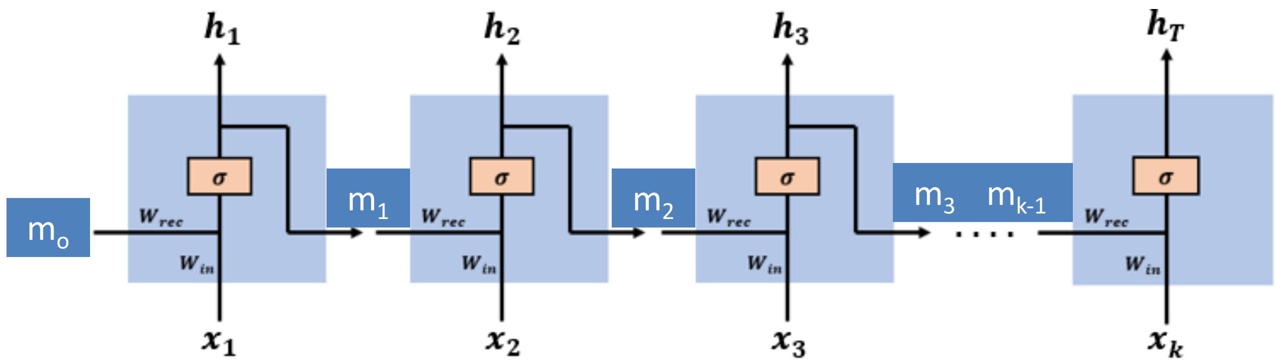
Problems of RNN

- The challenge of long-term dependencies
- Chain of many nonlinear functions
  - Most of the values associated with a tiny derivative
    ⇒ vanishing gradients
  - Some with a large derivative if the weights in W_h are large enough to overpower the small derivatives of the non-linear function ⇒ exploding gradients
  - Many alternate between increasing and decreasing
    ⇒ difficult to train

Some Maths

- Let's take a closer look!

$$\frac{\delta C_k}{\delta W_h} = \sum_{k=1}^{T} \frac{\delta C_k}{\delta W}$$



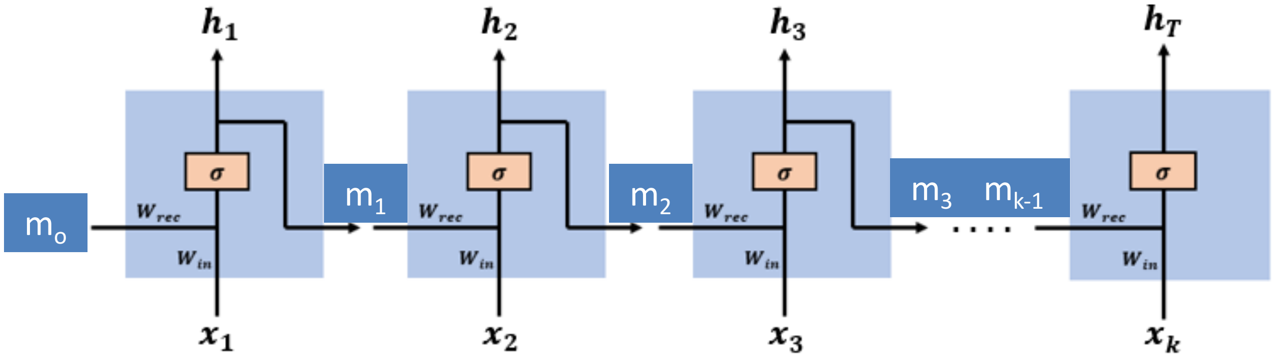
$$\frac{\delta C_k}{\delta W_h} = \frac{\delta C_k}{\delta h_k}\frac{\delta h_k}{\delta m_k} \cdots \frac{\delta m_2}{\delta m_1}\frac{\delta m_1}{\delta W}$$

$$= \frac{\delta C_k}{\delta h_k}\frac{\delta h_k}{\delta m_k}\left(\prod_{t=2}^{k}\frac{\delta m_t}{\delta m_{t-1}}\right)\frac{\delta m_1}{\delta W}$$

Some Maths

- Let's take a closer look!

$$\frac{\delta C_k}{\delta W_h} = \sum_{k=1}^{T} \frac{\delta C_k}{\delta W}$$



$$\frac{\delta C_k}{\delta W_h} = \frac{\delta C_k}{\delta h_k} \frac{\delta h_k}{\delta m_k} \cdots \frac{\delta m_2}{\delta m_1} \frac{\delta m_1}{\delta W}$$

$$= \frac{\delta C_k}{\delta h_k} \frac{\delta h_k}{\delta m_k} \left(\prod_{t=2}^{k} \frac{\delta m_t}{\delta m_{t-1}} \right) \frac{\delta m_1}{\delta W}$$

Some Maths

- Let's take a closer look!

$$m_t = f(W_h m_{t-1} + W_{in} x_t)$$

$$\frac{\delta m_t}{\delta m_{t-1}} = f'(m_{t-1}).W_h$$

Some Maths

- Let's take a closer look!

$$m_t = f(W_h m_{t-1} + W_{in} x_t)$$

$$\frac{\delta m_t}{\delta m_{t-1}} = f'(m_{t-1}).W_h$$

$$\frac{\delta C_k}{\delta W_h} = \frac{\delta C_k}{\delta h_k}\frac{\delta h_k}{\delta m_k}(\prod_{t=2}^{k}\frac{\delta m_t}{\delta m_{t-1}})\frac{\delta m_1}{\delta W}$$

$$= \frac{\delta C_k}{\delta h_k}\frac{\delta h_k}{\delta m_k}(\prod_{t=2}^{k} f'(m_{t-1}).W_h)\frac{\delta m_1}{\delta W}$$

Some Maths

- Let's take a closer look!

$$m_t = f(W_h m_{t-1} + W_{in} x_t)$$

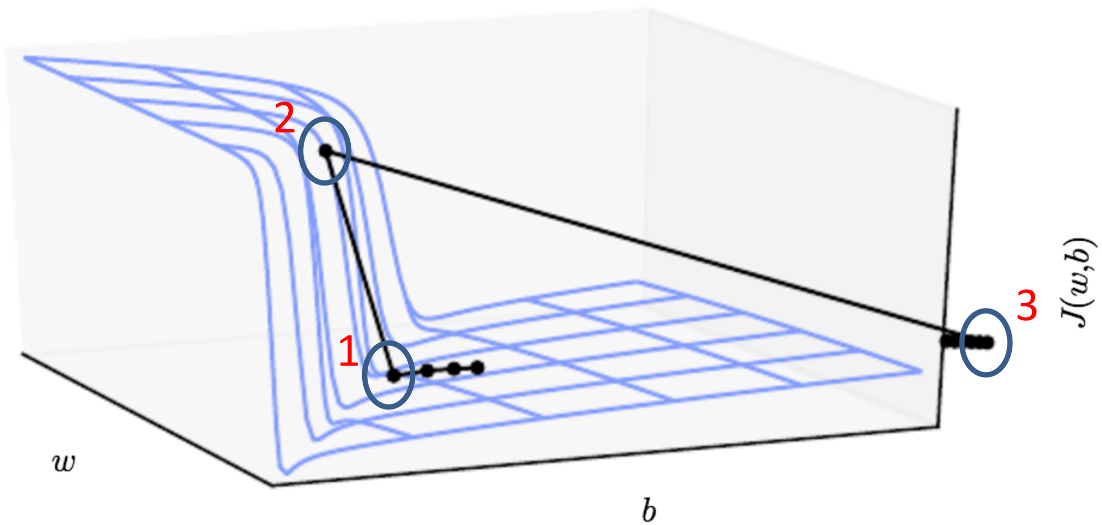
$$\frac{\delta m_t}{\delta m_{t-1}} = f'(m_{t-1}).W_h$$

$$\frac{\delta C_k}{\delta W_h} = \frac{\delta C_k}{\delta h_k}\frac{\delta h_k}{\delta m_k}\left(\prod_{t=2}^{k}\frac{\delta m_t}{\delta m_{t-1}}\right)\frac{\delta m_1}{\delta W}$$

$$= \frac{\delta C_k}{\delta h_k}\frac{\delta h_k}{\delta m_k}\left(\prod_{t=2}^{k} f'(m_{t-1}).W_h\right)\frac{\delta m_1}{\delta W}$$

Exploding Gradients

- In case of very deep networks or recurrent neural networks, gradients explode



Deep learning book

Gradient Clipping

- A simple heuristic solution is to clip the norm $\| g \|$ of the gradient g

$$\text{if } \| g \| > v :$$

$$g \leftarrow \frac{g \cdot v}{\| g \|}$$

Vanishing Gradients