Deep Learning

Vazgen Mikayelyan

YSU, Krisp

November 9, 2019

Outline

Sequential Data

- What is sequential data?

Sequential Data

- What is sequential data?
  - Time series

Sequential Data

- What is sequential data?
  - Time series
  - Text

Sequential Data

- What is sequential data?
  - Time series
  - Text
  - Speech

Sequential Data

- What is sequential data?
  - Time series
  - Text
  - Speech
  - Audio

Sequential Data

- What is sequential data?
  - Time series
  - Text
  - Speech
  - Audio
  - Video

Sequential Data

- What is sequential data?
  - Time series
  - Text
  - Speech
  - Audio
  - Video
- Why don't to use known networks?

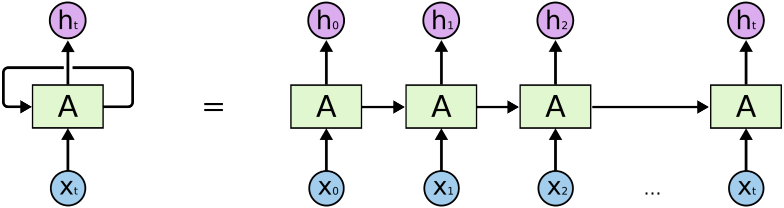
Sequential Data

- What is sequential data?
  - Time series
  - Text
  - Speech
  - Audio
  - Video
- Why don't to use known networks?
  - Inputs, outputs can be different lengths in different examples.

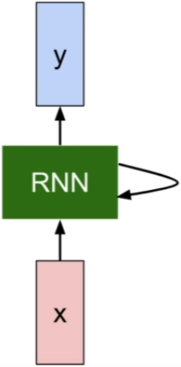
Sequential Data

- What is sequential data?
  - Time series
  - Text
  - Speech
  - Audio
  - Video
- Why don't to use known networks?
  - Inputs, outputs can be different lengths in different examples.
  - Doesn't share features learned across different positions of sequence.

Basic RNNs



Basic RNNs



$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

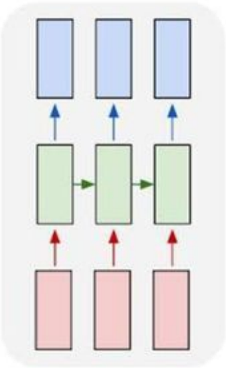
$$y_t = W_{hy}h_t$$

Different Types of RNNs

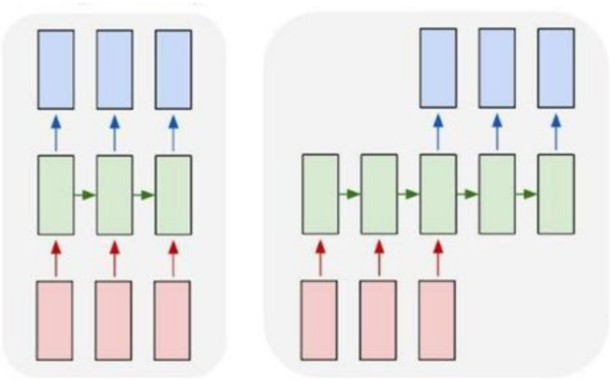
- Many to many

Different Types of RNNs

- Many to many



- Many to many

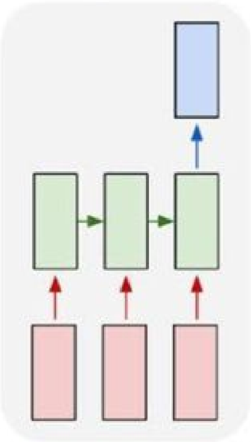


Different Types of RNNs

- Many to one

Different Types of RNNs

- Many to one

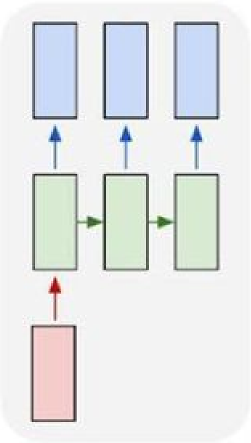


Different Types of RNNs

- One to many

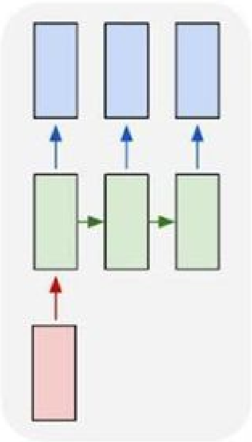
Different Types of RNNs

- One to many



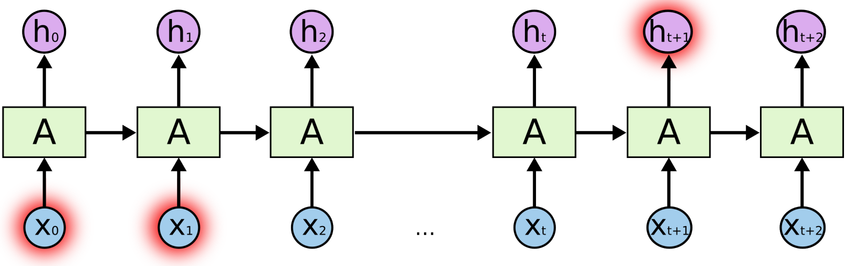
Different Types of RNNs

- One to many



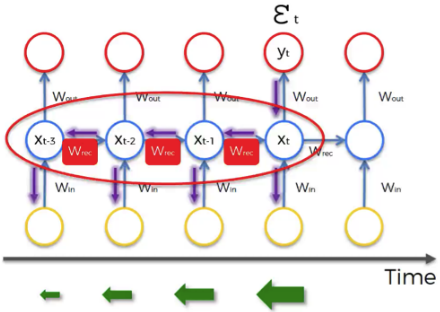
- One to one?

Problem of Long Term Dependencies



Problem of Vanishing Gradient

The Vanishing Gradient Problem



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

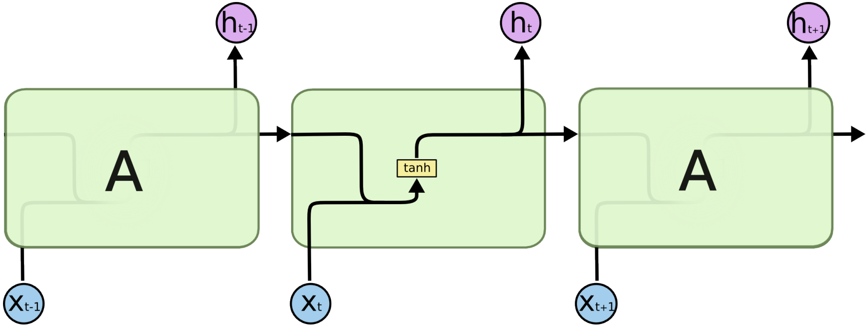
W_{rec} ~ small	⇨	Vanishing
W_{rec} ~ large	⇨	Exploding

Formula Source: Razvan Pascanu et al. (2013)

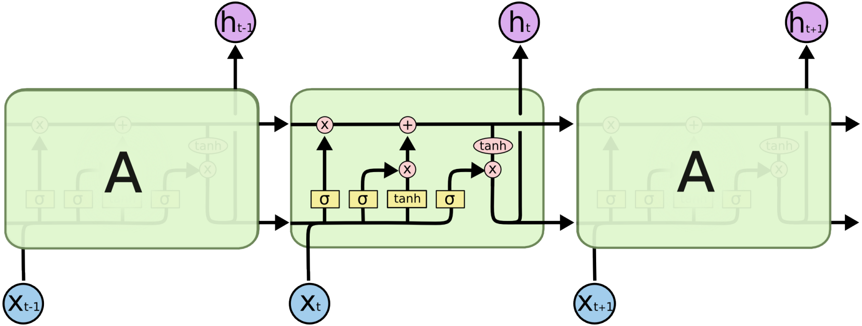
Deep Learning A-Z

Outline

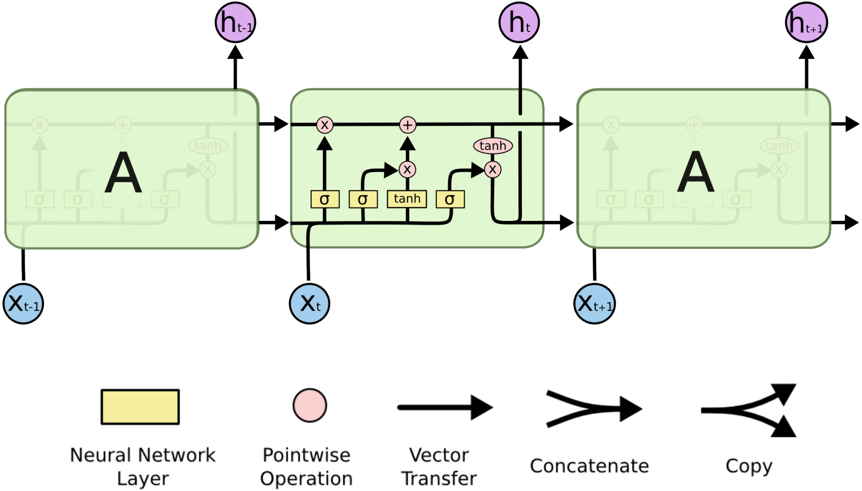
Simple RNN



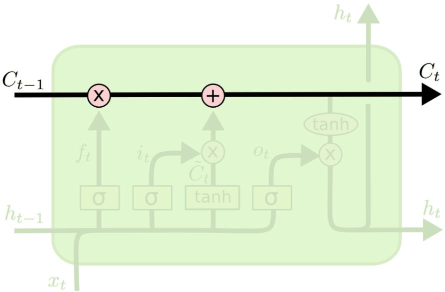
LSTM



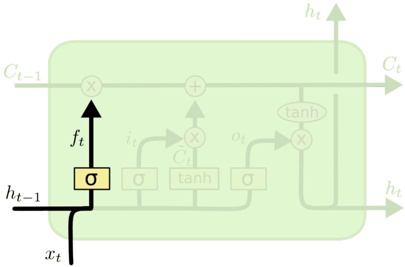
LSTM


h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

Additional state

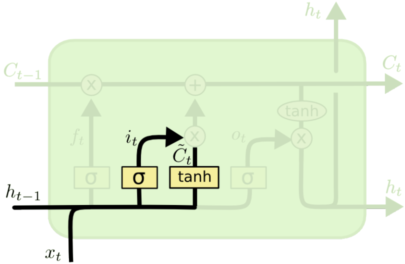


LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

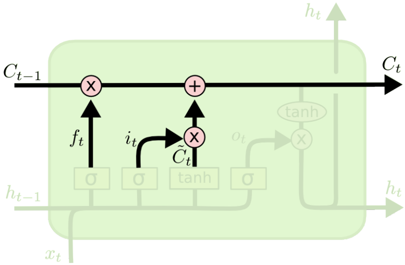
LSTM



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

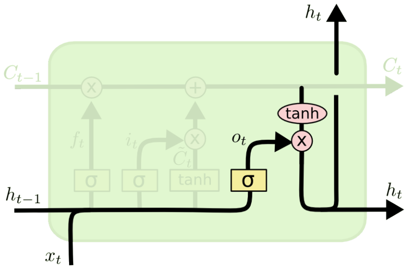
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

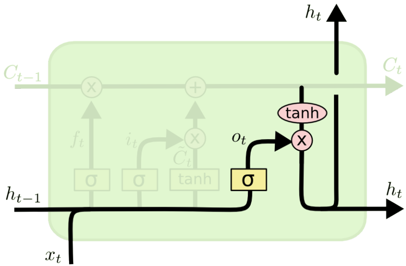
LSTM



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

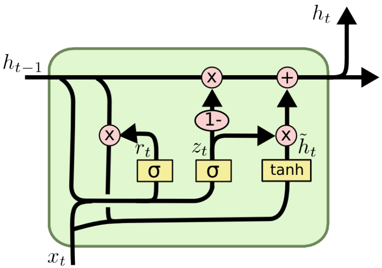
LSTM



$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Why to use tanh?

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

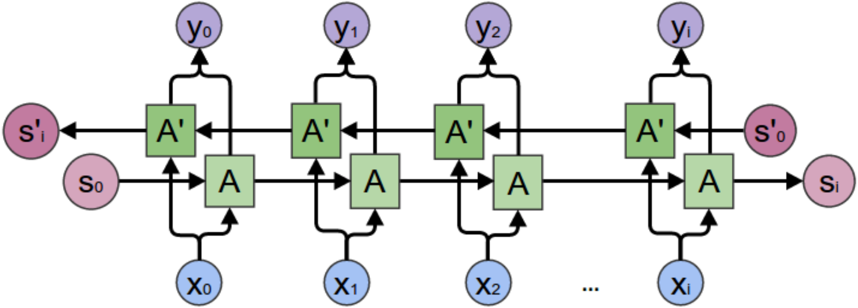
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

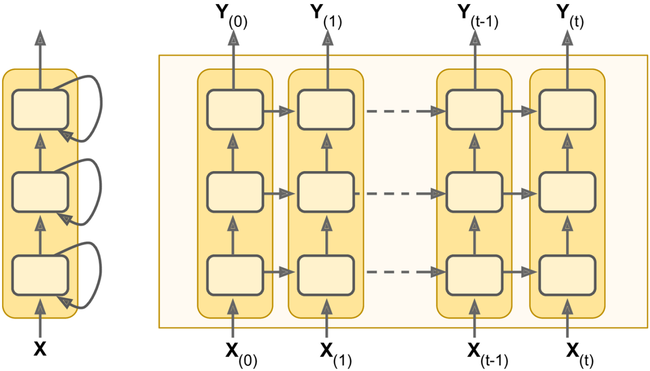
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Outline

BiRNN



Deep RNNs



Outline

Basic Idea

- Encode each word in the sentence into a vector using RNNs.

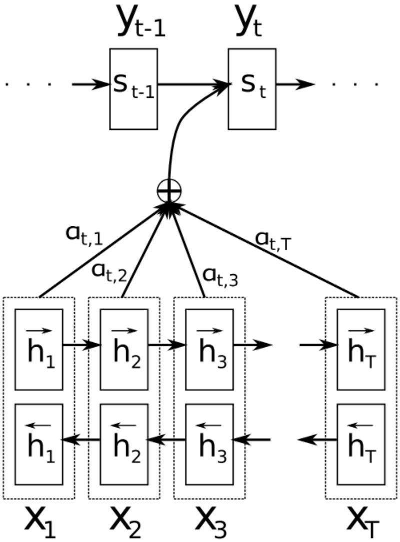
Basic Idea

- Encode each word in the sentence into a vector using RNNs.
- When decoding, perform a convex combination of these vectors, weighted by "attention weights".

Basic Idea

- Encode each word in the sentence into a vector using RNNs.
- When decoding, perform a convex combination of these vectors, weighted by "attention weights".
- Use this combination in picking the next word.

Attention Model



Attention Model

Input of decoder is a convex combination of encoded words:

$$c_i = \sum_{i=1}^{T_x} \alpha_{ij} h_j, i = 1, \ldots, T_y$$

Attention Model

Input of decoder is a convex combination of encoded words:

$$c_i = \sum_{i=1}^{T_x} \alpha_{ij} h_j, i = 1, \ldots, T_y$$

where

$$\alpha_{ij} = \frac{\exp\{e_{ij}\}}{\sum_{i=1}^{T_y} \exp\{e_{ij}\}}$$

Attention Model

Input of decoder is a convex combination of encoded words:

$$c_i = \sum_{i=1}^{T_x} \alpha_{ij} h_j, i = 1, \dots, T_y$$

where

$$\alpha_{ij} = \frac{\exp\{e_{ij}\}}{\sum_{i=1}^{T_y} \exp\{e_{ij}\}}$$

and $e_{i,j}$ are calculated using alignment model

$$e_{ij} = a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

Outline

Problems with RNNs

- Sequential computation prevents parallelization.

Problems with RNNs

- Sequential computation prevents parallelization.
- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length for codependent computation between states grows with sequence.

Problems with RNNs

- Sequential computation prevents parallelization.
- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length for codependent computation between states grows with sequence.
- But if attention gives us access to any state, maybe we don't need the RNN?

Transformer