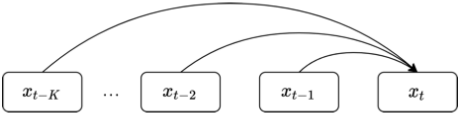
Recurrent Neural Networks (RNNs)

COMS4995 Fall 2021 Tutorial 7

Slides by: Arpit Bahety

Recap

Autoregressive methods: Predict next data observation as a linear equation of previously observed data points.



- Ex: $x_t = w_1 * x_{t-1} + w_2 * x_{t-2} + \ldots + w_K * x_{t-K}$
- Representational ability is limited. Only looks K steps back in time!

Recurrent Neural Networks (RNNs) offer several advantages:

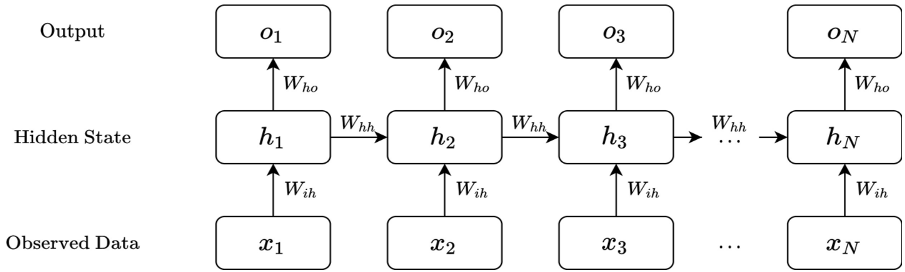
- Can represent long term dependencies in hidden state (theoretically).
- Shared weights, can be used on sequences of arbitrary length.

Overview of today's tutorial

1. RNN
    1. Applications
    2. Types
    3. Modifications
2. PyTorch example of RNN
3. GRU and LSTM
4. PyTorch example of LSTM

Most of the figures are obtained from the Deep Learning Specialization course by DeepLearning.AI. Haven't cited on each slide for the sake of brevity. I assume no ownership for these figures.

Recurrent Neural Networks



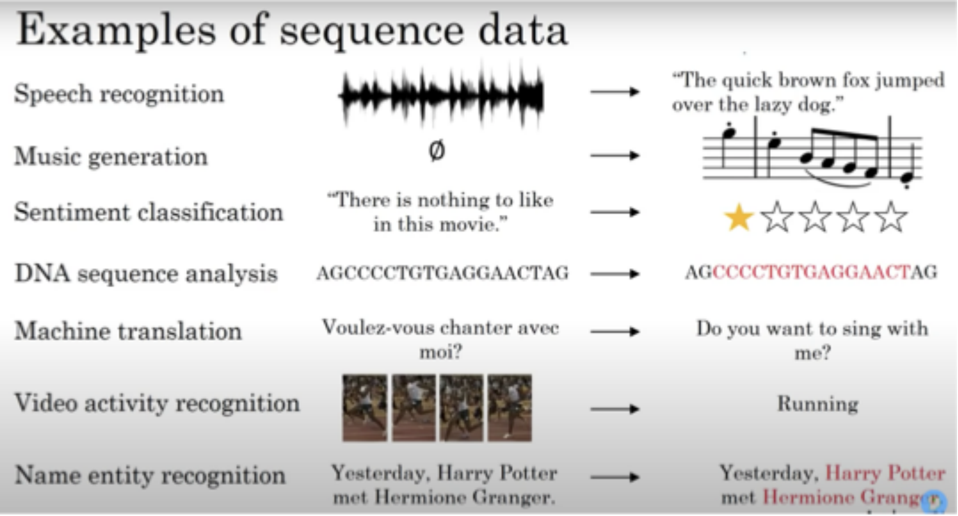
$$\mathbf{h_t} = W_{ih}\,\mathbf{x_t} + W_{hh}\,\boldsymbol{a}_{\mathbf{t-1}} + b_{ih} + b_{hh} \tag{1}$$

$$\mathbf{a_t} = \tanh(\mathbf{h_t}) \tag{2}$$

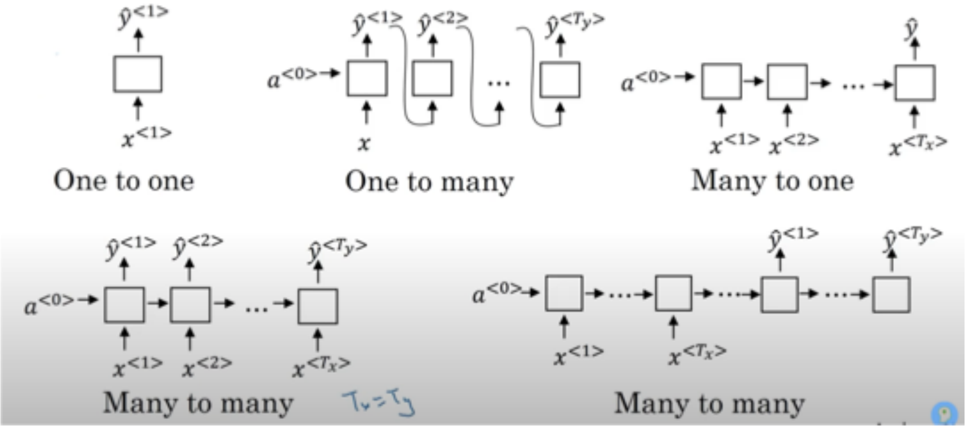
$$\mathbf{o_t} = \mathrm{softmax}(W_{ho}\,\mathbf{a_t} + b_{ho}) \tag{3}$$

Weight matrices are shared, meaning sequence can be arbitrary length.

Applications of RNN



Types of RNN

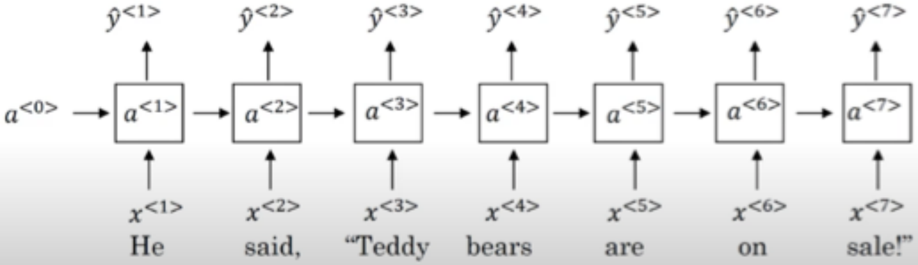

One to one
One to many
Many to one
Many to many
$T_x = T_y$
Many to many

RNN Modifications: Bidirectional RNNs

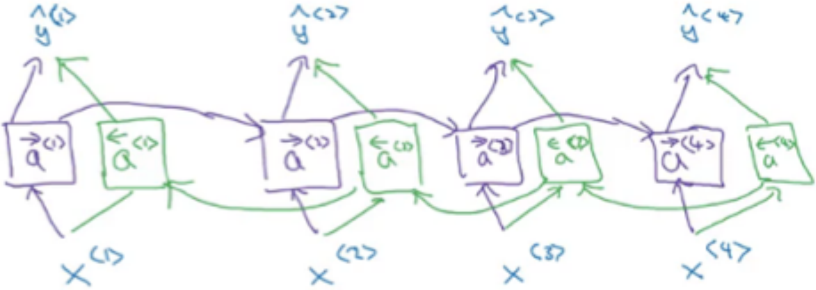
Getting information from the future

He said, “Teddy bears are on sale!”

He said, “Teddy Roosevelt was a great President!”



RNN Modifications: Bidirectional RNNs

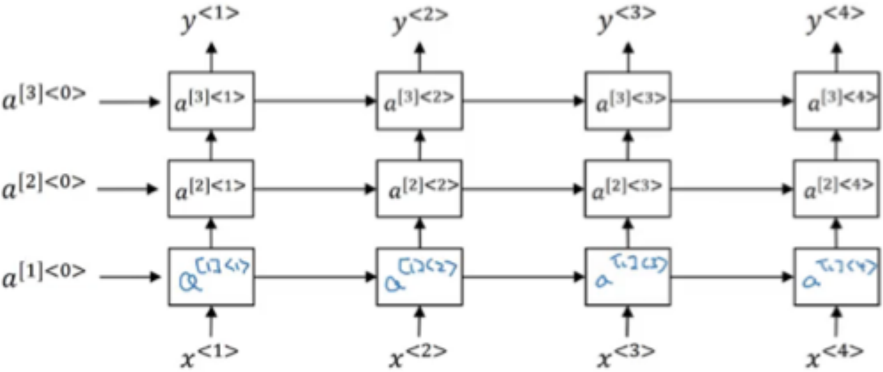


$$\hat{y}^{<t>} = g(W_y[\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

Runs two separate RNN in opposite directions, and concatenate output.

- Access to the future values can improve RNN representations.
- Disadvantage: Need the entire sequence before you can process it

RNN Modifications: Deep RNNs



$$a^{[2]<3>} = g\left(W_a^{[2]}\left[a^{[2]<2>}, a^{[1]<3>}\right] + b_a^{[2]}\right)$$

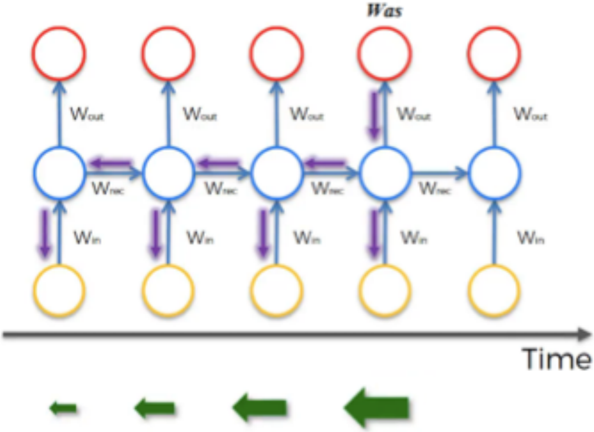
Live Demo: RNN Example

Switching to code notebook.

Questions?

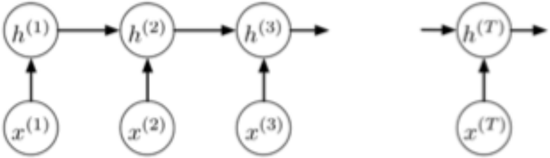
Gradient Issues

- The **cat**, which already ate ..., **was** full
- The **cats**, which already ate ..., **were** full



Recap from the lecture

Consider a univariate version of the encoder network:



Backprop updates:

$$\overline{h^{(t)}} = \overline{z^{(t+1)}}\, w$$

$$\overline{z^{(t)}} = \overline{h^{(t)}}\, \phi'(z^{(t)})$$

Applying this recursively:

$$\overline{h^{(1)}} = \underbrace{w^{T-1}\phi'(z^{(2)})\cdots\phi'(z^{(T)})}_{\text{the Jacobian } \partial h^{(T)}/\partial h^{(1)}}\,\overline{h^{(T)}}$$

With linear activations:

$$\partial h^{(T)}/\partial h^{(1)} = w^{T-1}$$

Exploding:

$$w = 1.1,\ T = 50 \quad \Rightarrow \quad \frac{\partial h^{(T)}}{\partial h^{(1)}} = 117.4$$

Vanishing:

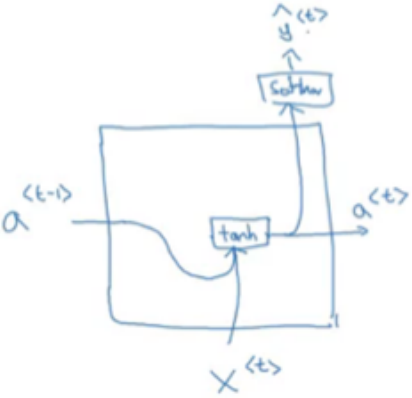
$$w = 0.9,\ T = 50 \quad \Rightarrow \quad \frac{\partial h^{(T)}}{\partial h^{(1)}} = 0.00515$$

GRU and LSTM

Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) units introduce long term cell state, allowing gradients to flow without being forced to change.

- Well, that description was unclear. Lets break it down!
- We'll start with GRU

RNN



tanh

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

GRU

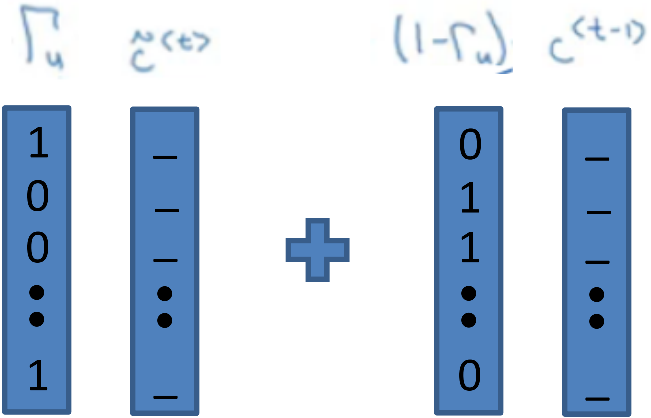
$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$$

Element-wise

Intuition of gates



Intuition of gates

Sentence: The **cat**, which already ate ...., **was** full

Word	Update gate(U)	Cell memory (C)
The	0	val
cat	1	new_val
which	0	new_val
already	0	new_val
...	0	new_val
was	1 (I don't need it anymore)	newer_val
full	..	..

Full GRU

- New gate that is used with to calculate the candidate C.
- The gate tells you how relevant is $C^{<t-1>}$ to $C^{<t>}$

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

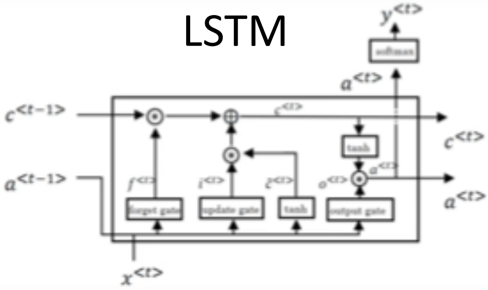
$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

LSTM



Live Demo: LSTM Example

Switching to code notebook.

Next Topics

Having to somehow pass long term information through hidden states may be a fundamentally flawed paradigm.

- Example: When we read, we don't actually look at the whole sentence, only keywords.

Next time on COMS4995 Attention & Transformers – teaching our models to focus on the important parts.