# COURSE 5: SEQUENCE MODELS

## Week 1

Examples of sequence data:

1. Speech recognition:  → "Hello, hi"
2. Music generation: ∅ or genre → 
3. Sentiment classification: "This movie sucks" → ★☆☆☆☆
4. DNA sequence analysis: AGCCCCTGTGAGGAACTAG → AGCCCCTGTGAGGAACTAG
5. Machine Translation: Voulez-vous chanter avec moi? → Do you want to sing with me?
6. Video Activity recognition:  → Running
7. Name Entity recognition: Yesterday, Harry met Hermione → Harry, Hermione

Notation:

$x$: Harry Potter and Hermione invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ . . . $x^{<t>}$ . . . . . $x^{<9>}$

$T_x = 9$

$y$: 1 1 0 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ . . . $y^{<t>}$ . . . $y^{<9>}$

$T_y = 9$

For training example $i$, $x^{(i)<t>}$, $y^{(i)<t>}$, $T_x^{(i)}$, $T_y^{(i)}$

The words are represented as vectors

Vocabulary

| | |
|---|---|
| a | 1 |
| aaron | 2 |
| ⋮ | |
| and | 367 |
| ⋮ | |
| harry | 4075 |
| ⋮ | |
| potter | 6830 |
| ⋮ | |
| zulu | 10,000 |

Then $x^{<1>}$ $x^{<2>}$ ← one-hot representation

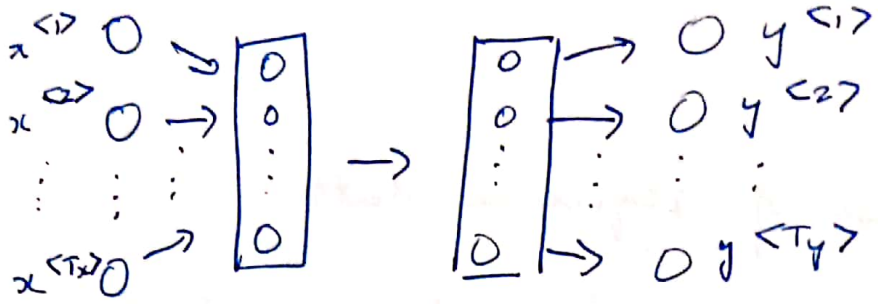
$x^{<1>}$: 0, 0, ⋮, 1, ⋮, 0 — 4075

$x^{<2>}$: ⋮, 1, ⋮, 0 — 6830

10,000

and so on for all 9 words

Why not a standard network?



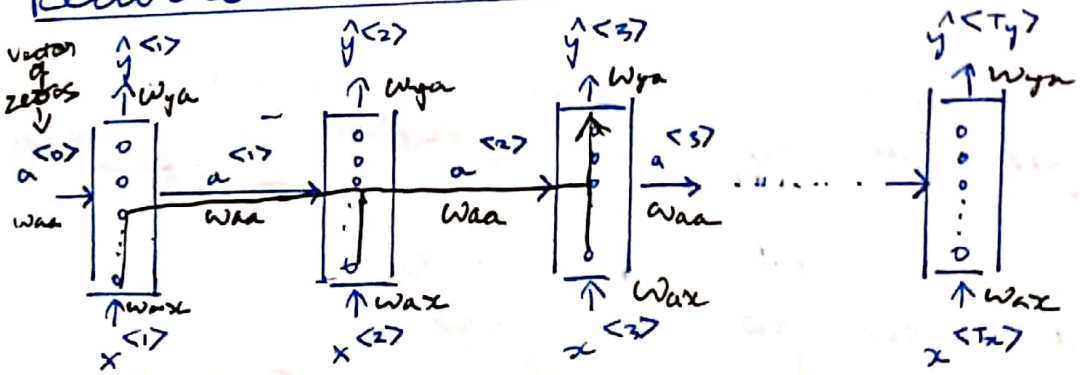

Problems:

→ Inputs, outputs can be different lengths in different examples (since the length of sentences vary).

→ Doesn't share features learned across different positions of text (It can't share features learnt if Harry comes at different places in different sentences)

So solution is RNN.

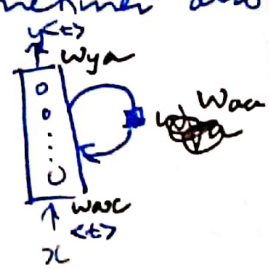
## Recurrent Neural Network




→ The parameters are shared

→ Activations of previous words affects the prediction of the current word (as shown by black path)

→ This is sometimes also represented as:



→ Problem: It only uses information from previous words (example: it doesn't use 4, 5, 6... $T_x$ in the diagram (for $\hat{y}^{<3>}$))

name | He said, "Teddy | Roosevelt was a great President."

not a name | He said, "Teddy | bears are on sale!"

So we can't determine by only knowing previous words, so we use Bidirectional RNN (BRNN)

Forward Propogation:

$$a^{<0>} = \vec{0}$$

---

$$a^{<1>} = g\left(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a\right)$$

← tanh is a common choice since it removes vanishing gradient problem. ReLu is used less often.

$$\hat{y}^{<1>} = g\left(W_{ya}\, a^{<1>} + b_y\right)$$

↑ sigmoid / softmax based on problem

---

$$a^{<t>} = g\left(W_{aa}\, a^{<t-1>} + W_{aa}\, x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(W_{ya}\, a^{<t>} + b_y\right)$$

This can be written as,

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

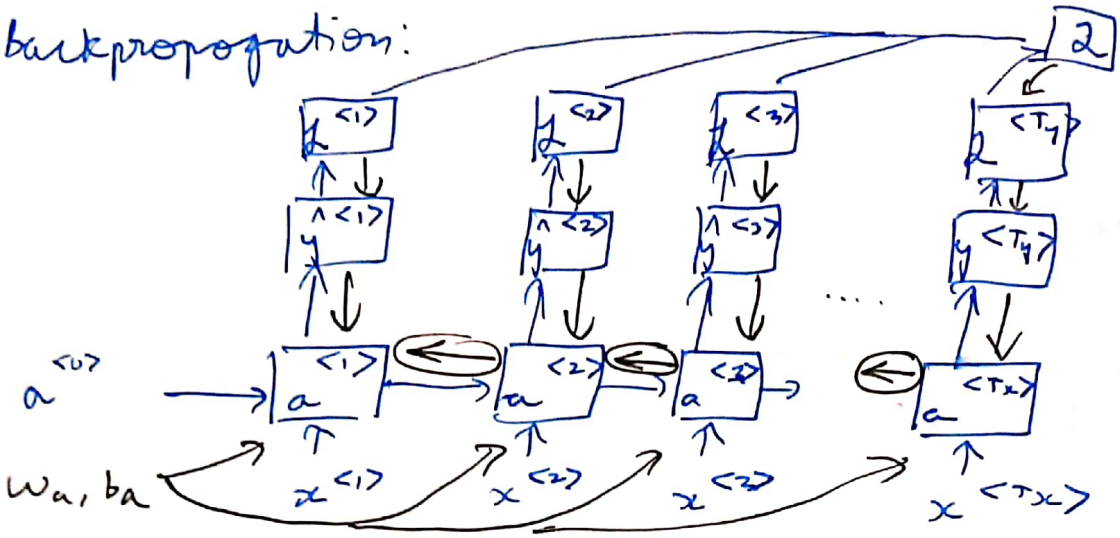
$$\hat{y}^{<t>} = g\left(W_y\, a^{<t>} + b_y\right)$$

If $W_{aa}$ is (100, 100) and $W_{ax}$ is (100, 10,000)

$$\underset{(100,\ 10{,}100)}{W_a} = \begin{array}{c} 100 \updownarrow \\ \\ \end{array}\left[\underbrace{W_{aa}}_{100} \vdots \underbrace{W_{ax}}_{10{,}000}\right] \text{ and } \underset{(10{,}100,\ 100)}{\left[a^{<t-1>}, x^{<t>}\right]} = \begin{array}{c} \overset{100}{\longleftrightarrow} \\ \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} \end{array} \begin{array}{l} \updownarrow 100 \\ \updownarrow 10{,}000 \end{array}$$

$$\therefore \left[W_{aa} \vdots W_{ax}\right]\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>}$$

backpropogation:




← is backpropogation

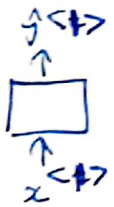
← is called backpropogation through time

$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>}\log \hat{y}^{<t>} - (1-y^{<t>})\log(1-\hat{y}^{<t>})$$

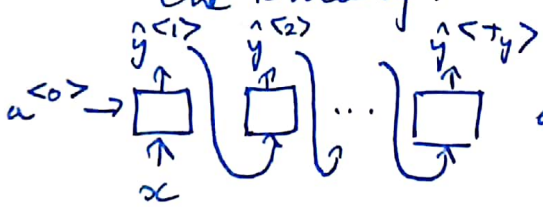
$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Different Types of RNN:

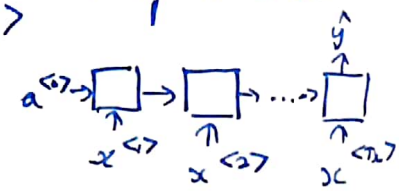
One-to-one:



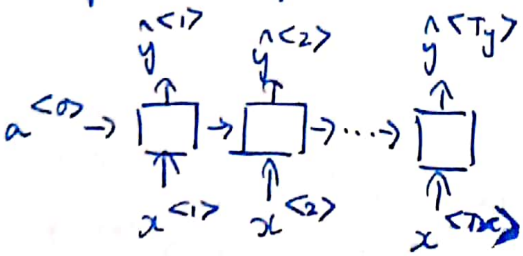

One to many:



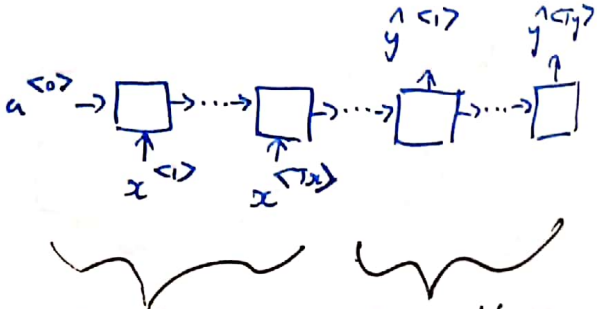

Many to one




Many to many:




Many to many:




Used in translation

What is language modelling?

Given a sentence, it'll tell you the probability of that sentence occuring.

1. The apple and pair salad $P = 3.2 \times 10^{-13}$

✓ 2. The apple and pear salad $P = 5.7 \times 10^{-10}$

$P(\text{sentence}) = P(y^{<1>}, y^{<2>}, \dots, y^{<T_y>}) = ?$

Language modelling with an RNN:

- Training set: large corpus of english text
- We tokenise the sentence first (convert to the number in the vocabulary list)
- Cats average 15 hours of sleep a day. <EOS> (end of sentence)

$y^{<1>}$ $y^{<2>}$ . . . $y^{<8>}$ $y^{<9>}$

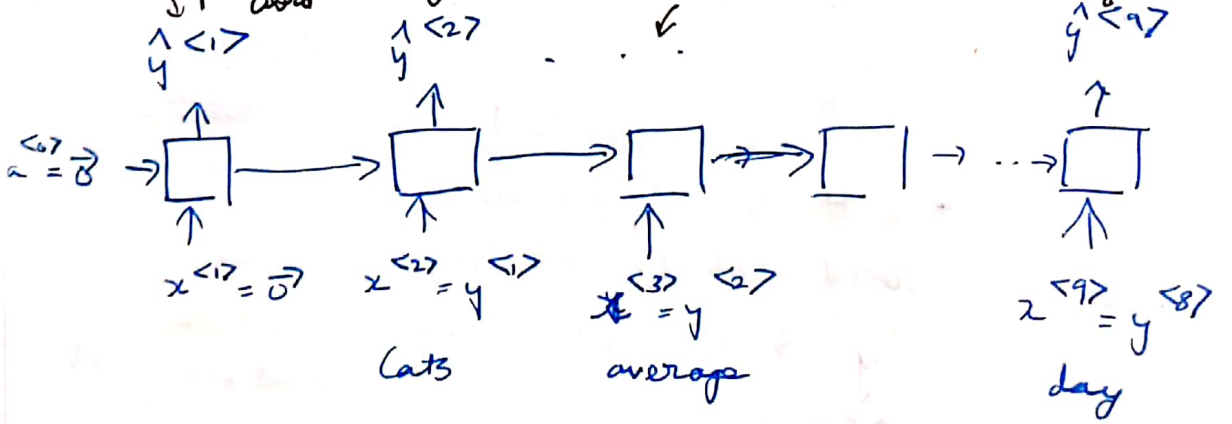
- The egyptian ~~Mau~~ <UNK> is a breed of cat. <EOS>

RNN model.

$P(\_ \mid \text{"cats"})$ ← what is the next word given cats

$P(\_ \mid \text{"Cats average"})$ ← what is the next word given cats average

Prediction of next word




$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$

$\mathcal{L} = \sum \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$

∴ Given a sentence

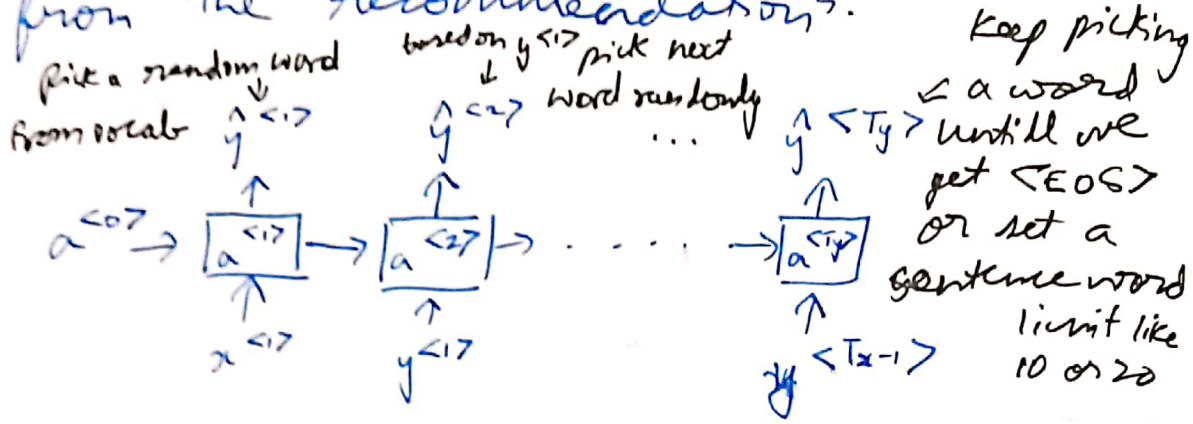
$P(y^{<1>}, y^{<2>}, y^{<3>}) =$

$P(y^{<1>}) P(y^{<2>} \mid y^{<1>})$

$P(y^{<3>} \mid y^{<1>}, y^{<2>})$

Sampling a sequence from a trained RNN:

Here we make is generate a sentence by making it randomly pick a word from the recommendations.




- Use np.random.choice to pick

- We can use a character level language model (with a vocabulary of characters) but it won't be good.

Vanishing gradients with RNNs:

If we have sentences like,

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

The cats, which . . . . . ., were full

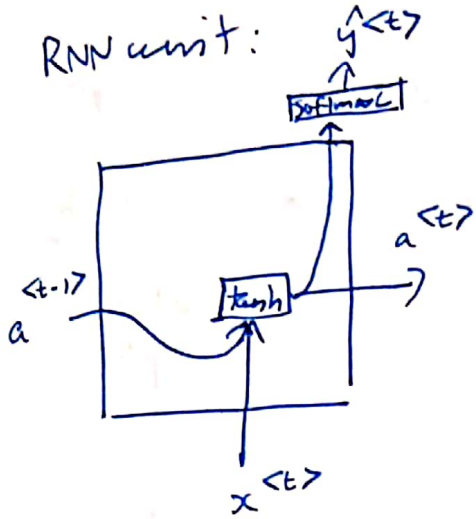
Since we saw in deeper NN, it's harder for a word at the beginning to affect a deeper word due to vanishing gradients. Usually only 3-4 words before have a strong influence on the current word using our current model. We will solve this.

- If exploding gradient occurs, we will get NaN errors. For this we need to do gradient clipping (setting a threshold for gradients so they don't get very large values)

# Gated Recurrent Unit (GRU):

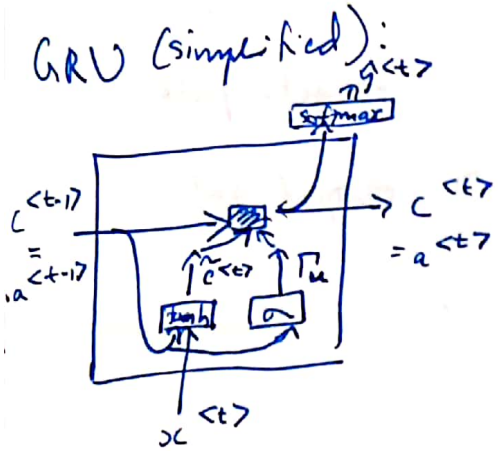
This helps with the vanishing gradient problem and makes it much better capturing long range connections.

RNN unit:




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

GRU (simplified):




$c$ = memory cell

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

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

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

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

$\Gamma_u = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$ .. ... $\Gamma_u = 1$

$c^{<t>} = 1$

The cat which already ate .... was full

-> The memory cell remembers that cat is singular till gate is activated again

-> The gate decides when the $c^{<t>}$ value changes, otherwise it won't affect it (since $\Gamma_u = 0$, $\Gamma_u * \tilde{c}^{<t>} = 0$)

Full GRU:

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

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

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

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

Long Short Term Memory (LSTM): ← better option

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

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

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

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

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

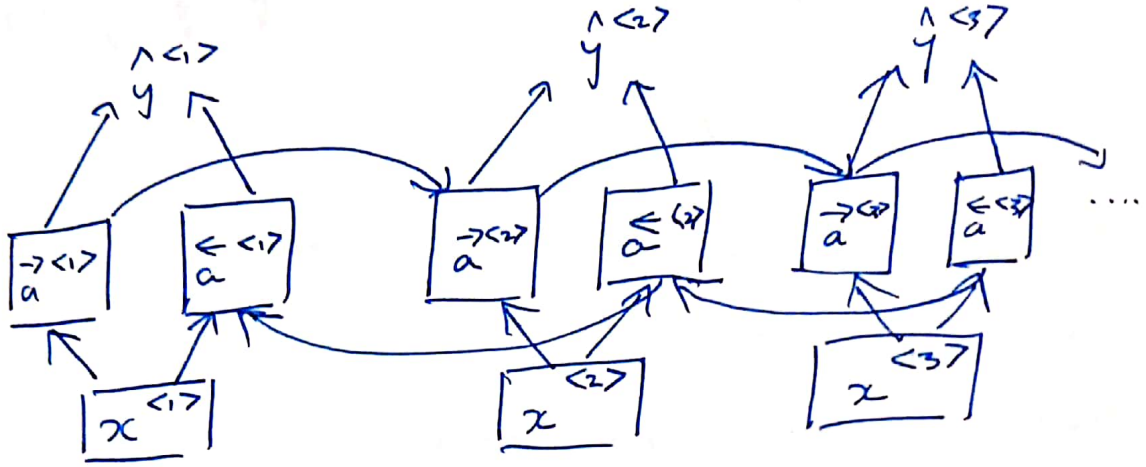
$\rightarrow a^{<t>} = \Gamma_o * \tanh(c^{<t>})$

Check video for diagram of LSTM

$y^{<t>}_{pred} = \text{softmax}(W_y a^{<t>} + b_y)$

# Bidirectional RNN (BRNN)

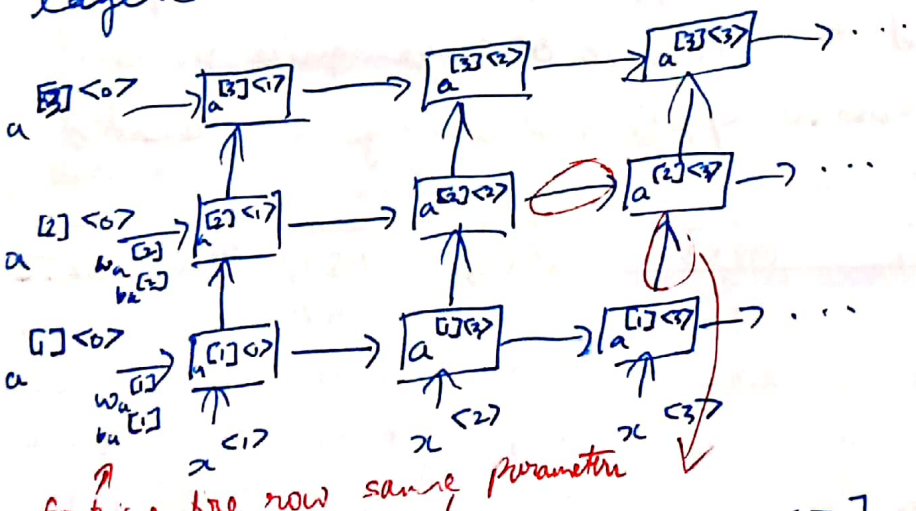
To address the issue of RNN using information only from the previous words, we use bidirectional RNN.




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

## Deep RNN example

You can make a deep RNN by stacking more layers horizontally.




For this entire row same parameter

Exa:

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