Recurrent Neural Network
↳ repeating

MLP x [vector] → MLP → y

CNN x [image] → CNN → y

Both of them do not take
advantage of sequences.

[This smartphone is fast & battery is good]
OR
[Battery good this is phone smart & is the]
} both are same for MLPs using word vectors

① words
② Time Series → stock market
③ Translation
④ Audio data → is a sequence
⑤ Image → caption
→ just to make it easier to index & search.

TLDR; if output depends on sequence, we need some form of RNN

Let's take an example of words here.

$x_i \rightarrow$ Google (x_1) will (x_2) kill (x_3) me (x_4) in (x_5) 2021 (x_6)

If we have one hot encoding for 10k words,
each word vector is 10k dim.

So if we have 5 words, we 50k dim if
we preserve order.

↓

The input is variable anyway. How do
you handle that?

Conclusion → This doesn't end up working.

Large word corpuses & large sequences
will always be a challenge for RNNs.

Review Sentiment Analysis

Dataset (D) = $\{x_i, y_i\}$
↳ binary

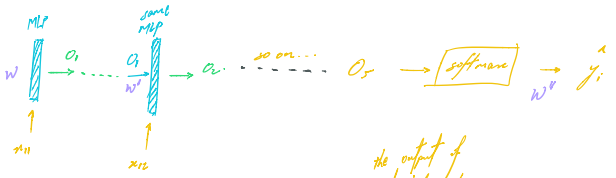
response

↓

will have
varying words
obviously

Train words → {0, 1}

$x_1 \rightarrow (x_{11}, x_{12}, x_{13}, x_{14}, x_{15})$



3 set of weights

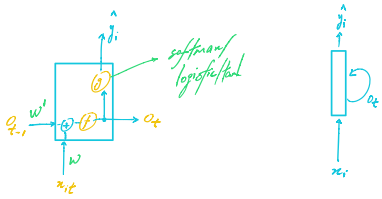
$W \rightarrow$ current input

$W' \rightarrow$ previous output

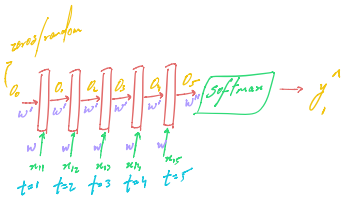
$W'' \rightarrow$ in the end

We want O_2 to depend on x_{12} & O_1

Each output depends on → current word + output of previous word.



Backprop over Time



$$a_1 = f(W x_{11} + W' a_0)$$

$$a_2 = f(W x_{12} + W' a_1)$$

$$\vdots$$

$$O_5 = f(W x_{15} + W' O_4)$$

$$\hat{y}_i = g(W'' \cdot O_5)$$

$$\frac{\partial L}{\partial W''} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial W''}$$

$$\frac{\partial L}{\partial (W)} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial O_5} \cdot \frac{\partial O_5}{\partial W}$$

W on the last time

$$\frac{\partial L}{\partial W'} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial O_5} \cdot \frac{\partial O_5}{\partial W'}$$

This backprop happens over time, on the same weights.

The longer the sequence, the larger the backprop product : Vanishing/Exploding gradient problem will persist.

↓

LSTM / GRU
avoid these problems through modifications to RNN

many to one.
↳ movie reviews, etc.

classification or regression both possible

one to many [image captioning]

image

many to many [same length]
part of speech detection → flag some particular word as verb or noun or a feature flag

many to many [different length]
↳ input & output

→ # translation tasks

Simple RNNs suck

① Vanishing gradients ruin both forward & backward passes
↳ if first & last word are related to each other, RNN will not effectively detect that.

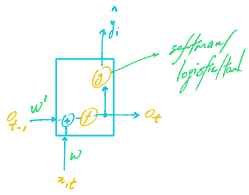
RNNs suck at long term dependencies in input sequences

see the distance between them

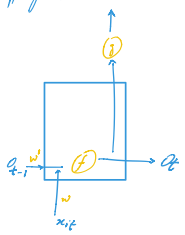
↳ we want the network to have complete input before outputting something

LSTM

#KID



#Legend



⇐ Vraiment si this is correct

$o_t = f([w', w], [x_{it}, o_{t-1}])$

concatenate (not add (like kirb))

& dot product

Colah's blog is great for LSTMs

GRU

→ optimized

→ less params to train

→ faster training

→ Almost same performances as LSTMs. ↗ special cases apply