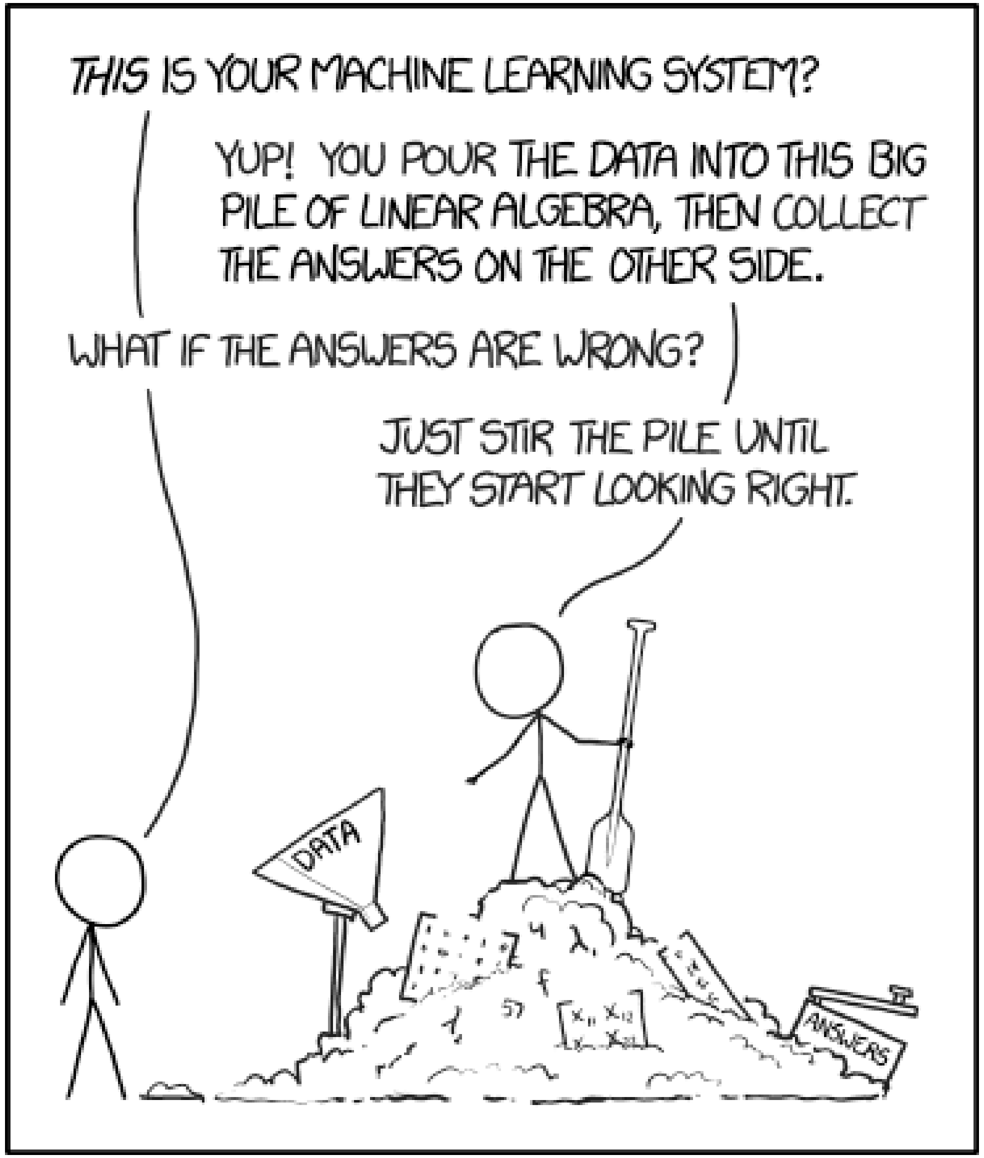
Introduction to Recurrent Neural Networks – Part 1

Devashish Khatwani
June 2018



A Motivating example for sequence modelling: Predict the next word

"This morning I took the dog for a walk."

"This morning I took the dog for a walk."

given these words

predict what comes next?

"This morning I took the dog for a walk."

given these 2 words, predict the next word

[1 0 0 0 0 0 1 0 0 0]

for a

One hot feature vector indicates what each word is

prediction

But...

"In France, I had a great time and I learnt some of the _____ language."

We need information from the far past and future to accurately guess the correct word.

Try using the whole sentence as a window!

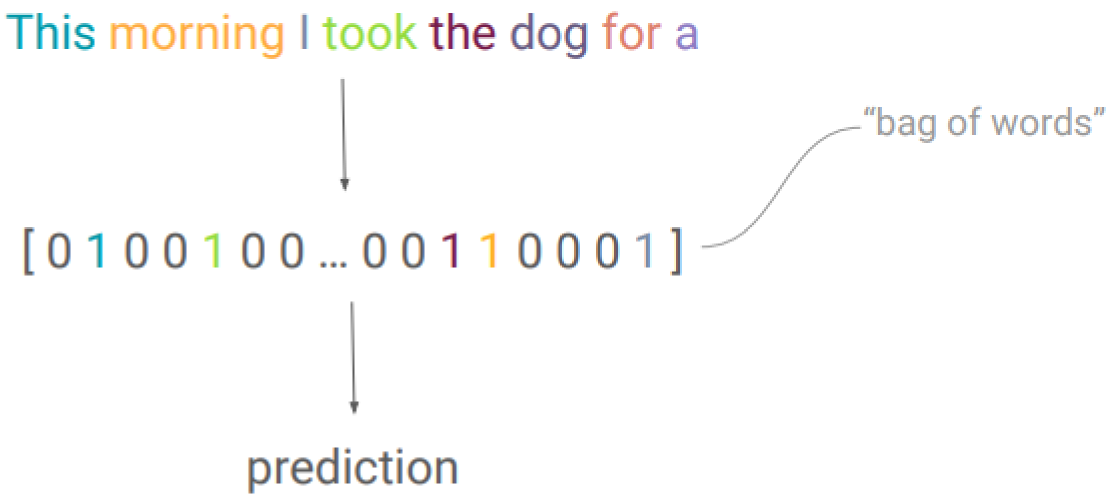
This morning I took the dog for a

[0 1 0 0 1 0 0 … 0 0 1 1 0 0 0 1]

"bag of words"

prediction

Try using the whole sentence as a window!



But… Counts don't preserve order

"The food was good, not bad at all."

vs

"The food was bad, not good at all."

Use really big windows!

"This morning I took the dog for a walk."

given these 7 words, predict the next word

[1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 ...]

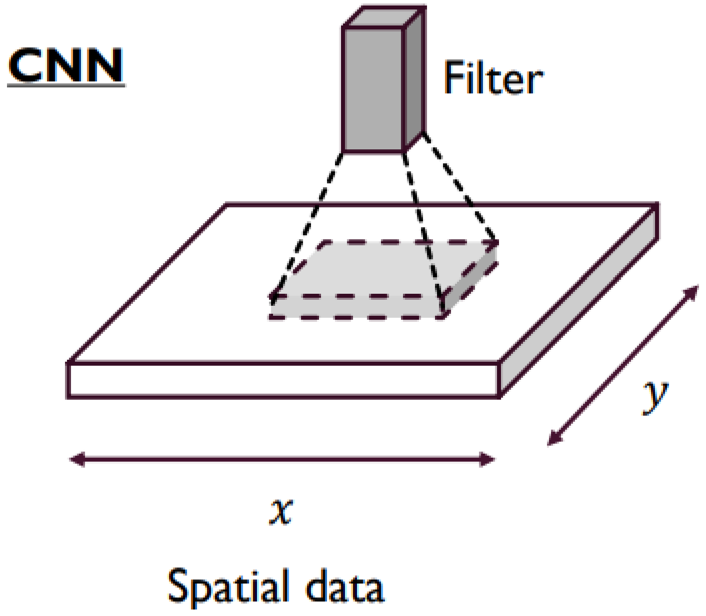
morning I took the dog ...

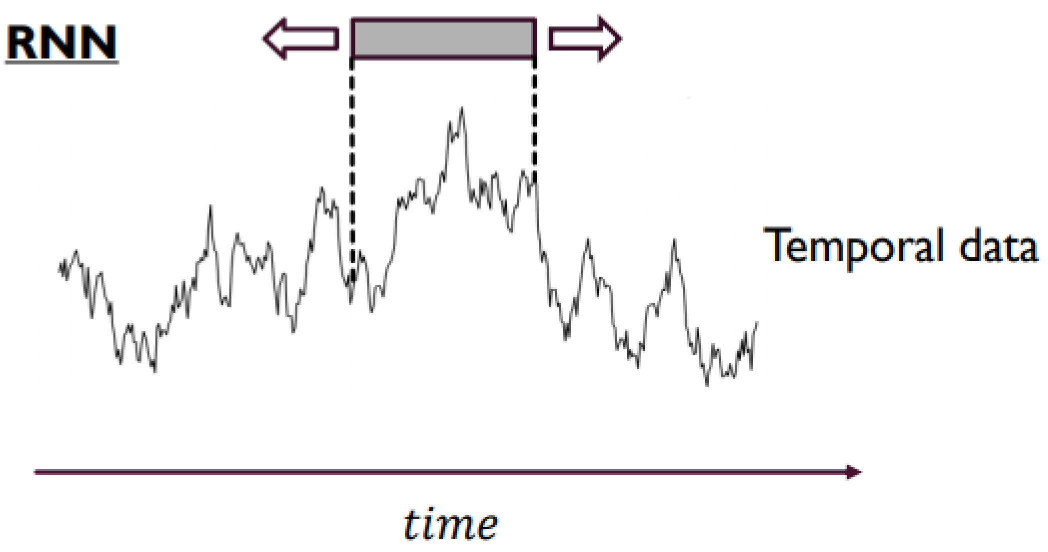
prediction

But... Curse of Dimensionality

Difference with CNNs

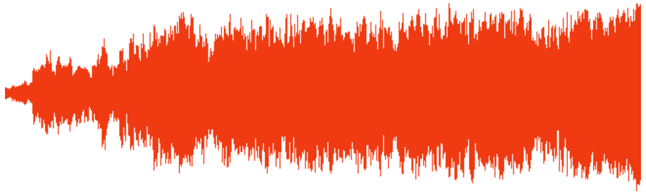
- Convolution in space (CNN) VS convolution in time (RNN)
- CNN: models relationships in space. Filter slides along x and y dimensions
- RNN: models relationships in time. ‘‘Filter’’ slides along time dimension





What is a Sequence?

1 Audio



2 ECG Reading

3 Sentence

Lucy is going to the park.

4 Bank Data?

To solve our problem we need…

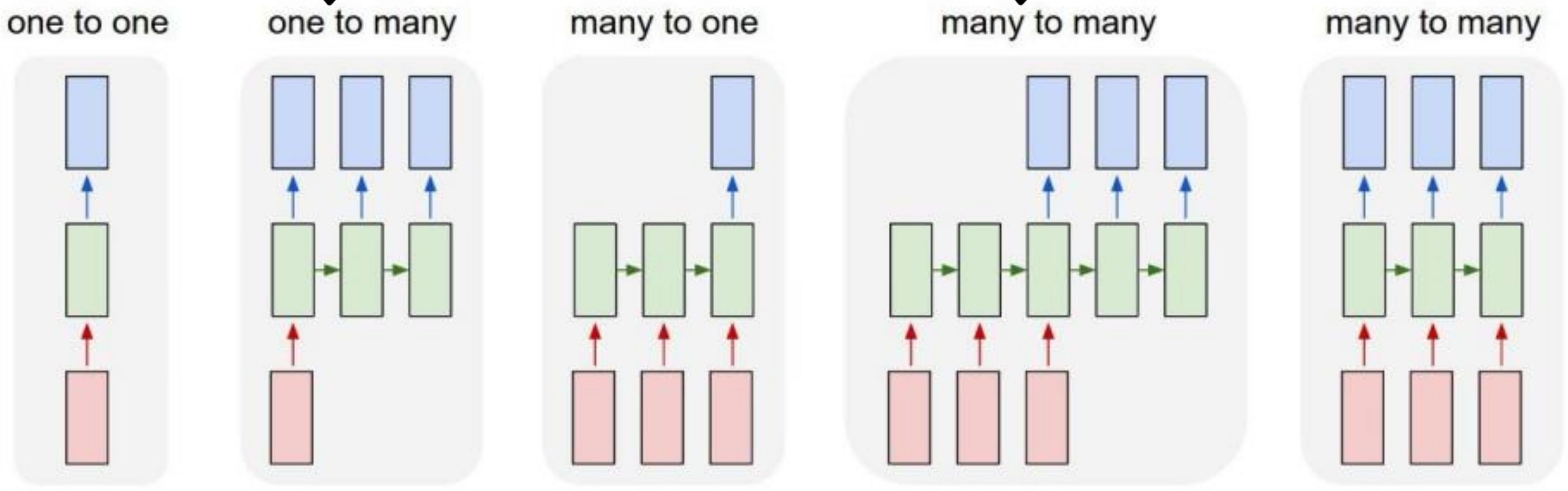
1. To deal with variable-length sequences
2. To maintain sequence order
3. To keep track of long-term dependencies
4. To share parameters across the sequence

Try out Recurrent Neural Networks

Types of RNNs

Image Captioning

Language Translation

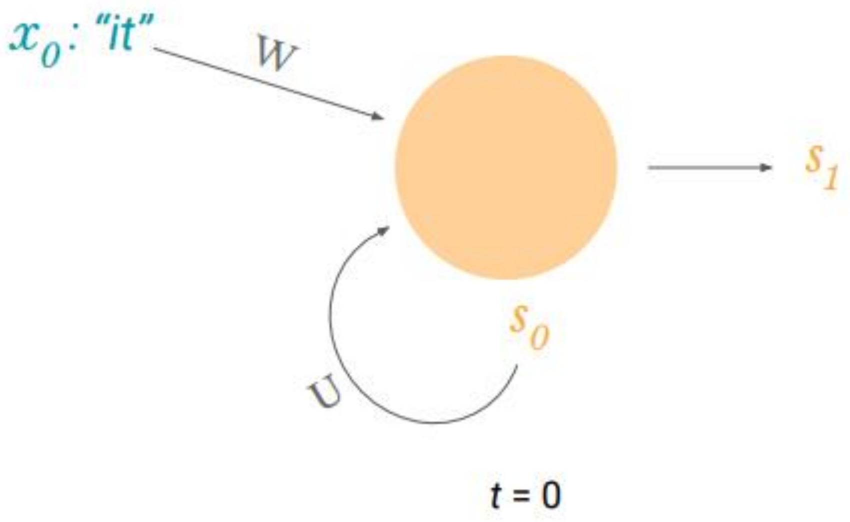


Not really a RNN!

Sentiment Classification

Video classification at frame level

RNNs remember their previous state

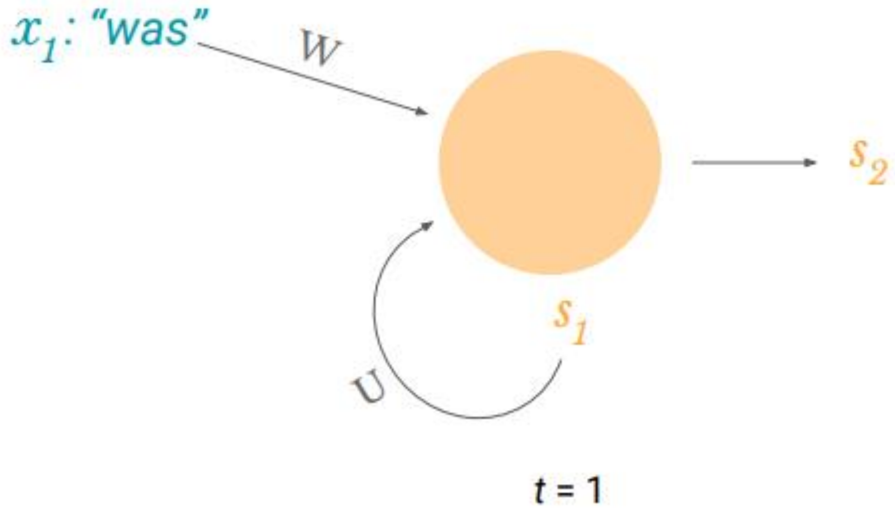


x_0 : vector representing first word
s_0 : cell state at $t = 0$ (some initialization)
s_1 : cell state at $t = 1$

$$s_1 = tanh(Wx_0 + Us_0)$$

$W, \ U$: weight matrices

RNNs remember their previous state

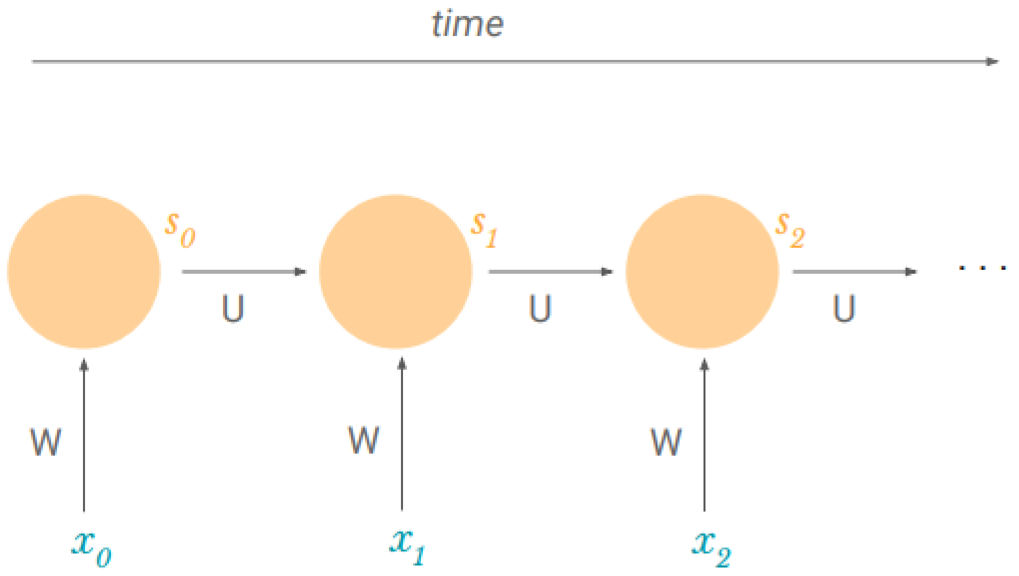


x_1 : vector representing second word
s_1 : cell state at $t = 1$
s_2 : cell state at $t = 2$

$$s_2 = tanh(Wx_1 + Us_1)$$

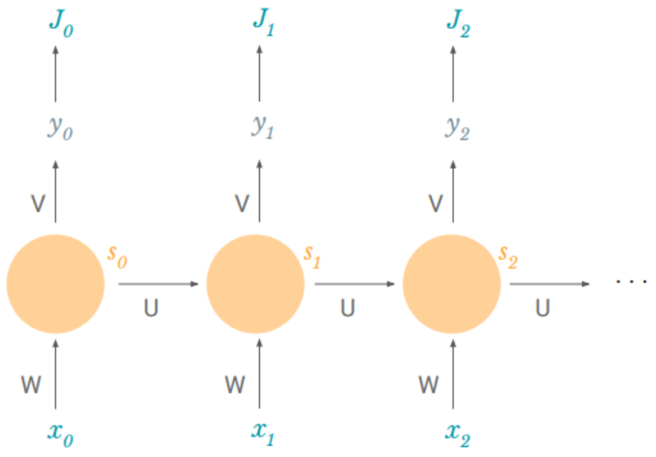
$W,\ U$: weight matrices

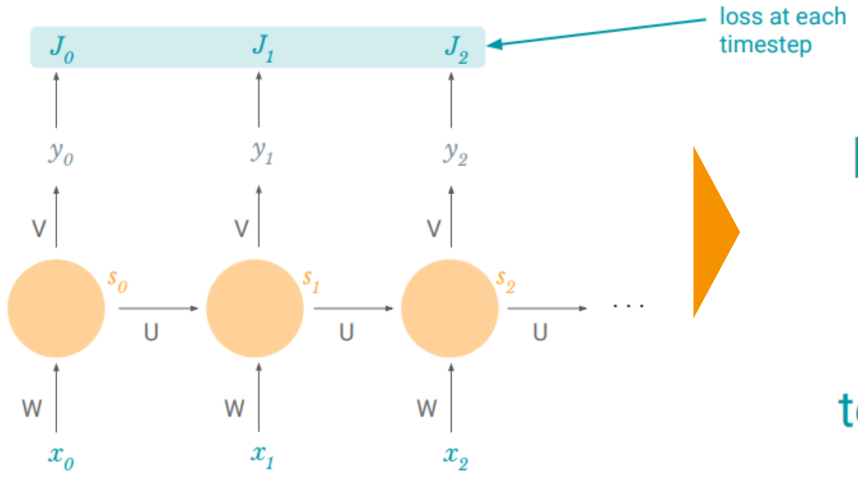
Unfolding the RNN across time



Notice that W and U are shared parameters across time

Training a RNN: What is our Loss?





loss at time t = $J_t(\boldsymbol{\Theta})$

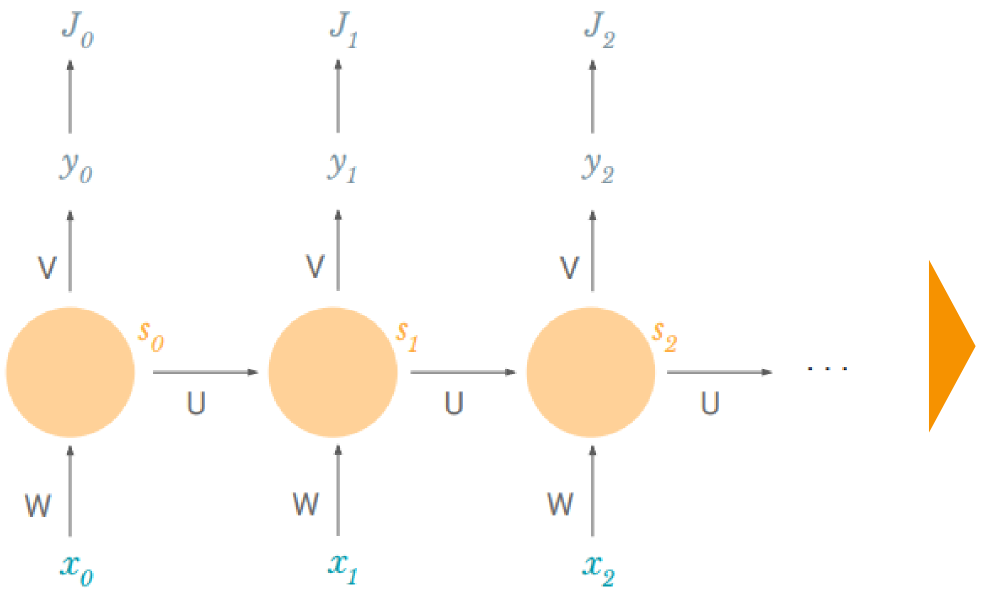
$\boldsymbol{\Theta}$ = our parameters, like weights

total loss = $J(\boldsymbol{\Theta}) = \Sigma_t \, J_t(\boldsymbol{\Theta})$

Training a RNN: What are our Gradients?

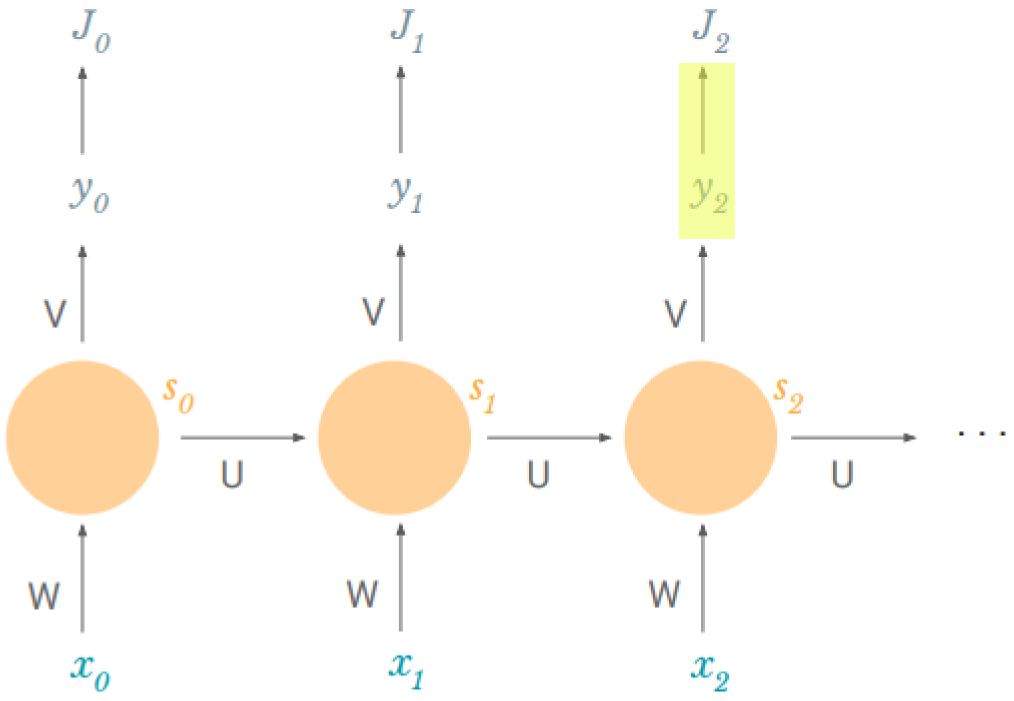
we sum gradients across time for each parameter *P*:

$$\frac{\partial J}{\partial P} = \sum_t \frac{\partial J_t}{\partial P}$$



$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

Training a RNN: Try it out with W

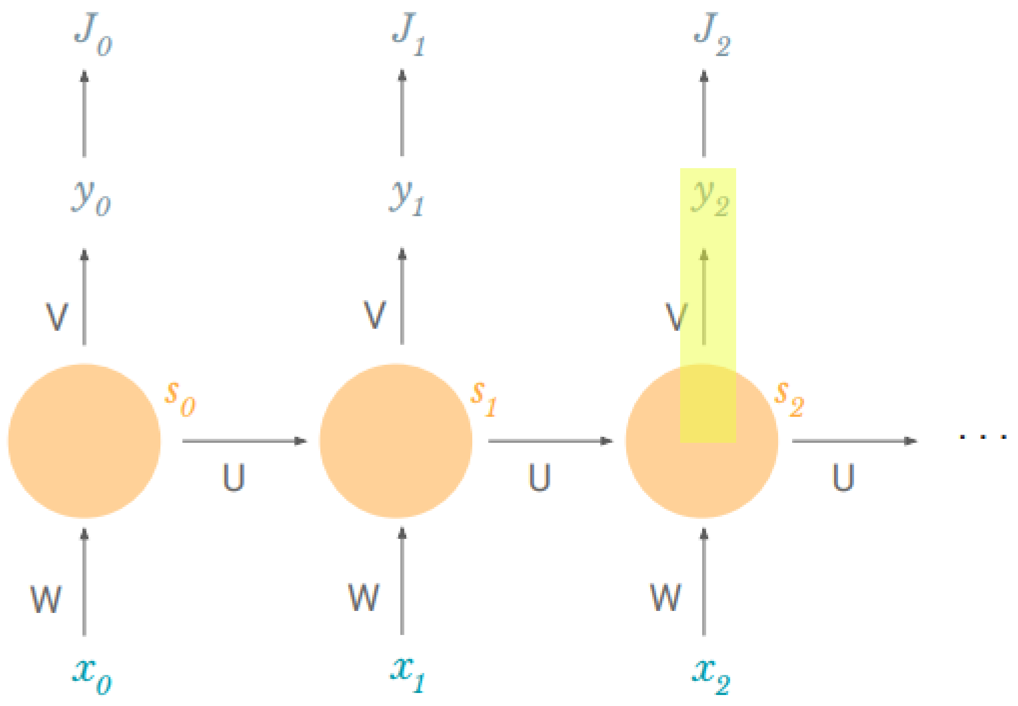


$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

so let's take a single timestep *t*:

$$\frac{\partial J_2}{\partial W} = \frac{\partial J_2}{\partial y_2}$$

Training a RNN: Try it out with W

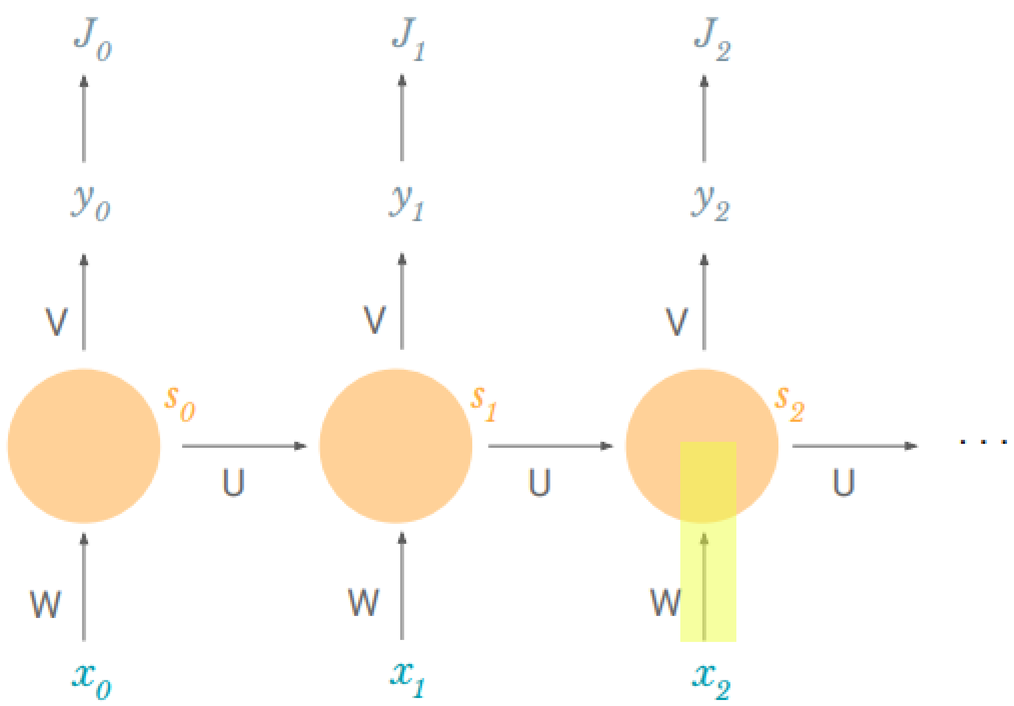


$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

so let's take a single timestep *t*:

$$\frac{\partial J_2}{\partial W} = \frac{\partial J_2}{\partial y_2}\frac{\partial y_2}{\partial s_2}$$

Training a RNN: Try it out with W

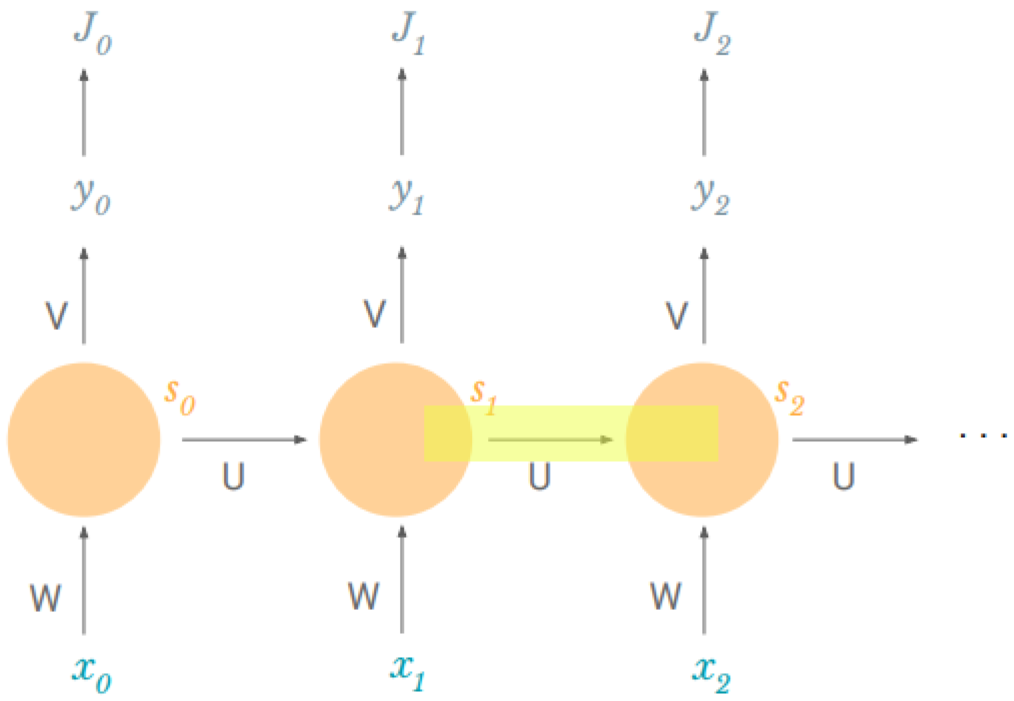


$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

so let's take a single timestep *t*:

$$\frac{\partial J_2}{\partial W} = \frac{\partial J_2}{\partial y_2} \frac{\partial y_2}{\partial s_2} \frac{\partial s_2}{\partial W}$$

Training a RNN: Try it out with W



$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

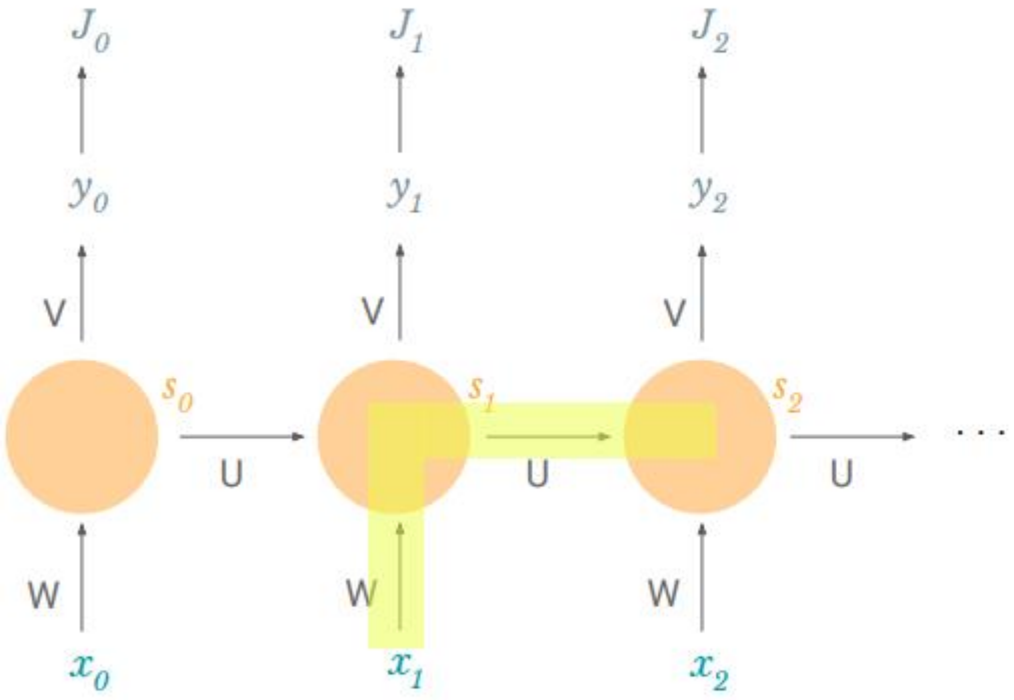
so let's take a single timestep *t*:

$$\frac{\partial J_2}{\partial W} = \frac{\partial J_2}{\partial y_2}\frac{\partial y_2}{\partial s_2}\frac{\partial s_2}{\partial W}$$

but wait...

$$s_2 = tanh(Us_1 + Wx_2)$$

Training a RNN: Try it out with W



$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

so let's take a single timestep *t*:

$$\frac{\partial J_2}{\partial W} = \frac{\partial J_2}{\partial y_2} \frac{\partial y_2}{\partial s_2} \boxed{\frac{\partial s_2}{\partial W}}$$

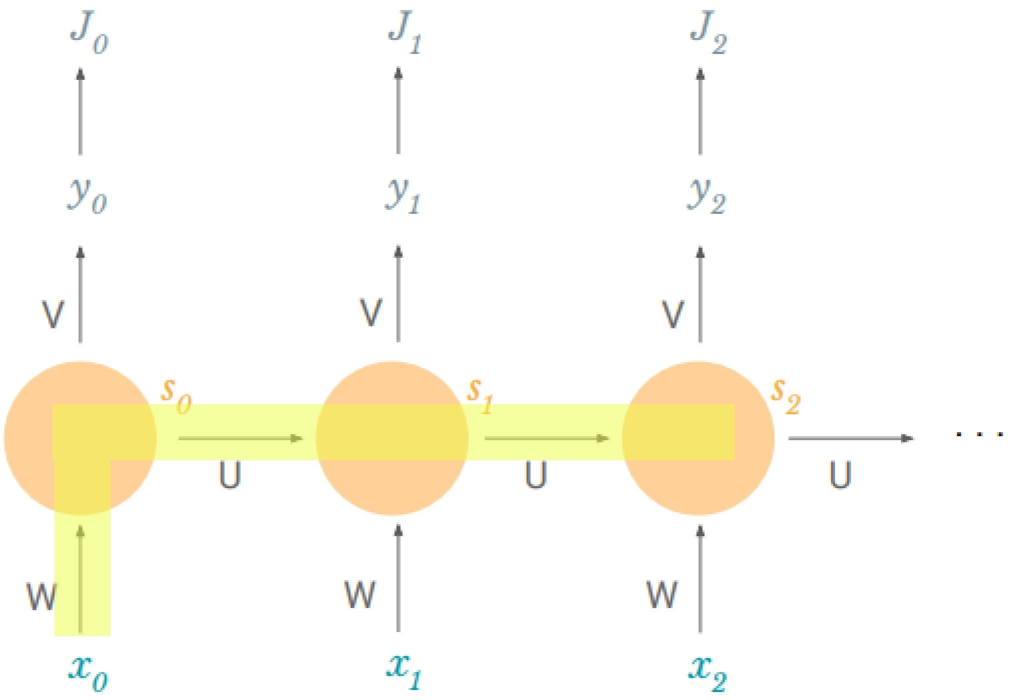
but wait...

$$s_2 = tanh(U\boxed{s_1} + Wx_2)$$

s_1 also depends on *W* so we can't just treat $\frac{\partial s_2}{\partial W}$ as a constant!

Training a RNN: Try it out with W

how does s_2 depend on W?



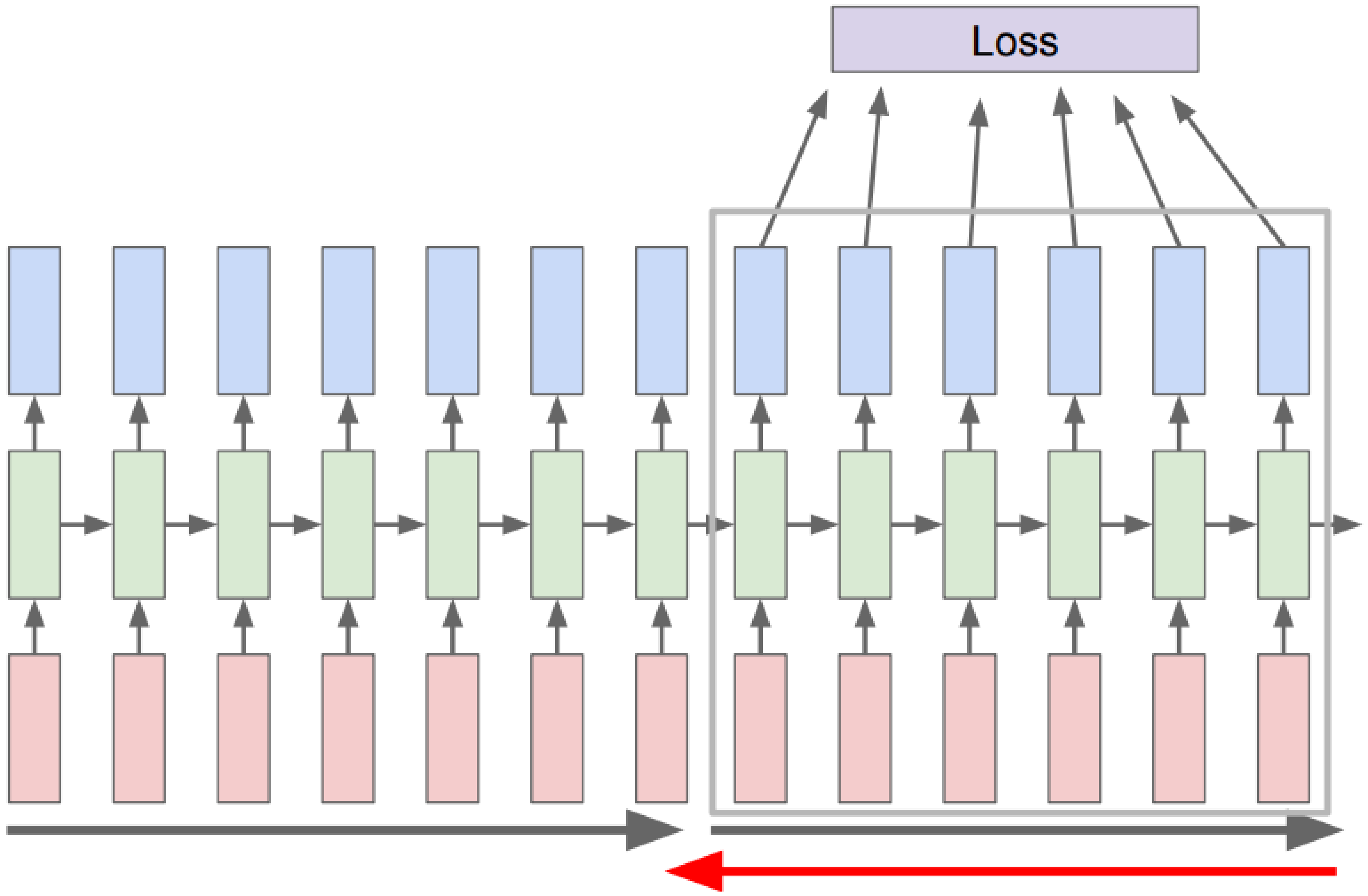
$$\frac{\partial s_2}{\partial W} + \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W} + \frac{\partial s_2}{\partial s_0}\frac{\partial s_0}{\partial W}$$

Training a RNN: Backpropagation Through Time

$$\frac{\partial J_2}{\partial W} = \sum_{k=0}^{2} \frac{\partial J_2}{\partial y_2} \frac{\partial y_2}{\partial s_2} \underbrace{\frac{\partial s_2}{\partial s_k} \frac{\partial s_k}{\partial W}}_{\substack{\text{Contributions of } W \text{ in previous} \\ \text{timesteps to the error at timestep } t}}$$

$$\frac{\partial J_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial J_t}{\partial y_t} \frac{\partial y_t}{\partial s_t} \underbrace{\frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W}}_{\substack{\text{Contributions of } W \text{ in previous} \\ \text{timesteps to the error at timestep } t}}$$

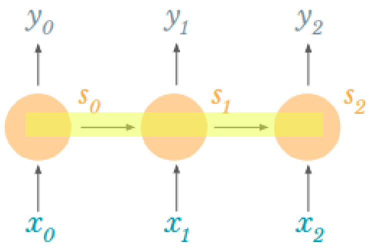
Training a RNN: Truncated Backpropagation Through Time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

But RNNs are hard to train: Vanishing Gradient

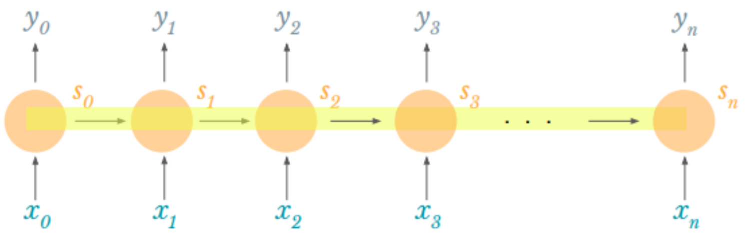
$$\frac{\partial J_2}{\partial W} = \sum_{k=0}^{2} \frac{\partial J_2}{\partial y_2} \frac{\partial y_2}{\partial s_2} \boxed{\frac{\partial s_2}{\partial s_k}} \frac{\partial s_k}{\partial W}$$



at k = 0: $\boxed{\frac{\partial s_2}{\partial s_0} = \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial s_0}}$

$$\frac{\partial J_n}{\partial W} = \sum_{k=0}^{n} \frac{\partial J_n}{\partial y_n} \frac{\partial y_n}{\partial s_n} \boxed{\frac{\partial s_n}{\partial s_k}} \frac{\partial s_k}{\partial W}$$

$$\boxed{\frac{\partial s_n}{\partial s_{n-1}} \frac{\partial s_{n-1}}{\partial s_{n-2}} \cdots \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial s_0}}$$



Two Hacky Solutions

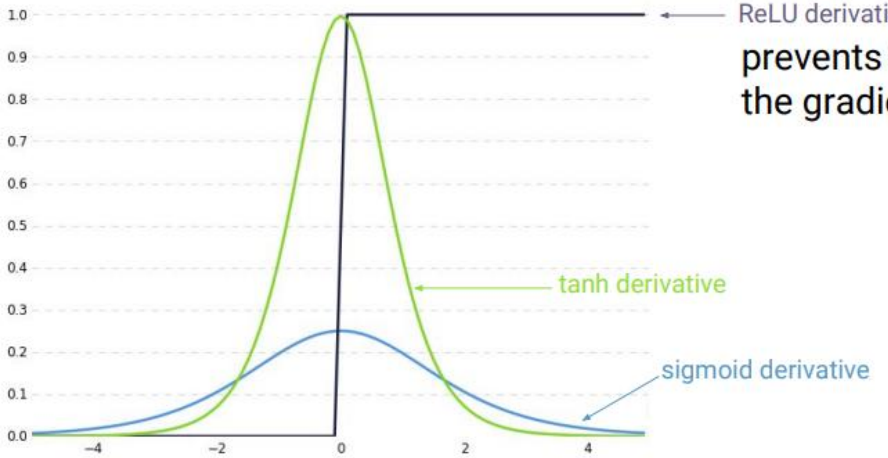
what are each of these terms? → $\frac{\partial s_n}{\partial s_{n-1}} \frac{\partial s_{n-1}}{\partial s_{n-2}} \cdots \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial s_0}$

$$\frac{\partial s_n}{\partial s_{n-1}} = W^T diag[f'(W_{s_{j-1}} + Ux_j)]$$

W = sampled from standard normal distribution = mostly < 1

f = tanh or sigmoid so **f'** < 1

1



prevents f' from shrinking the gradients

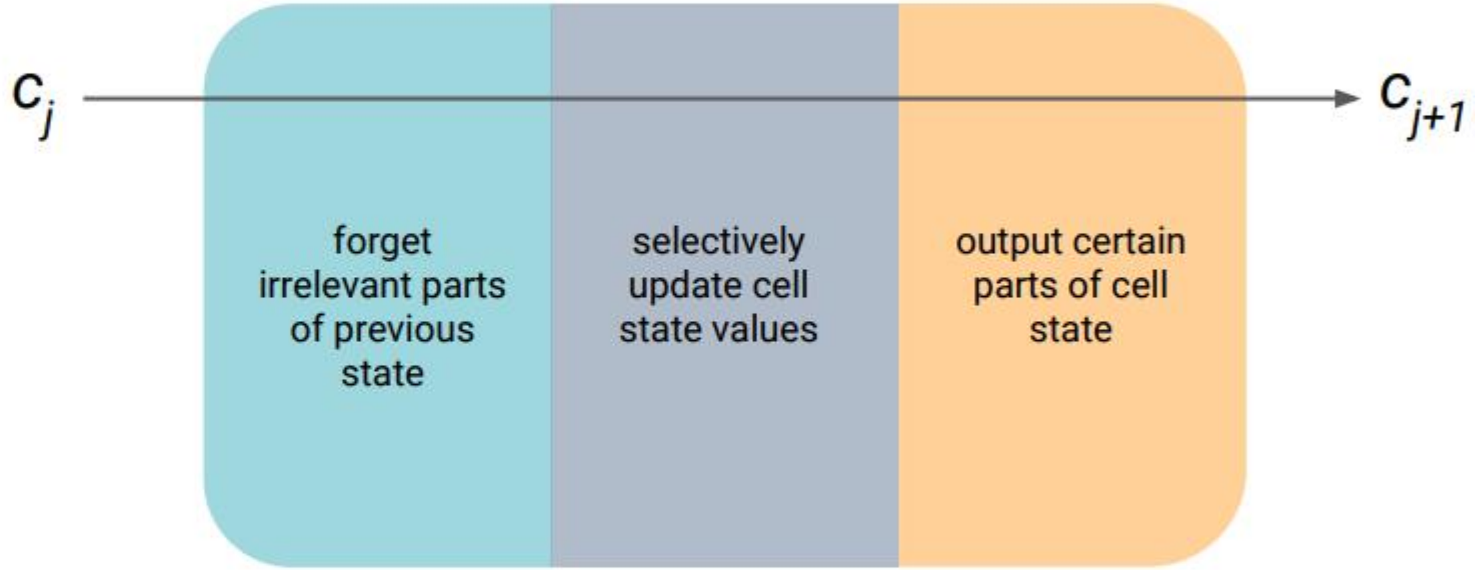
2

weights initialized to identity matrix → $I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$

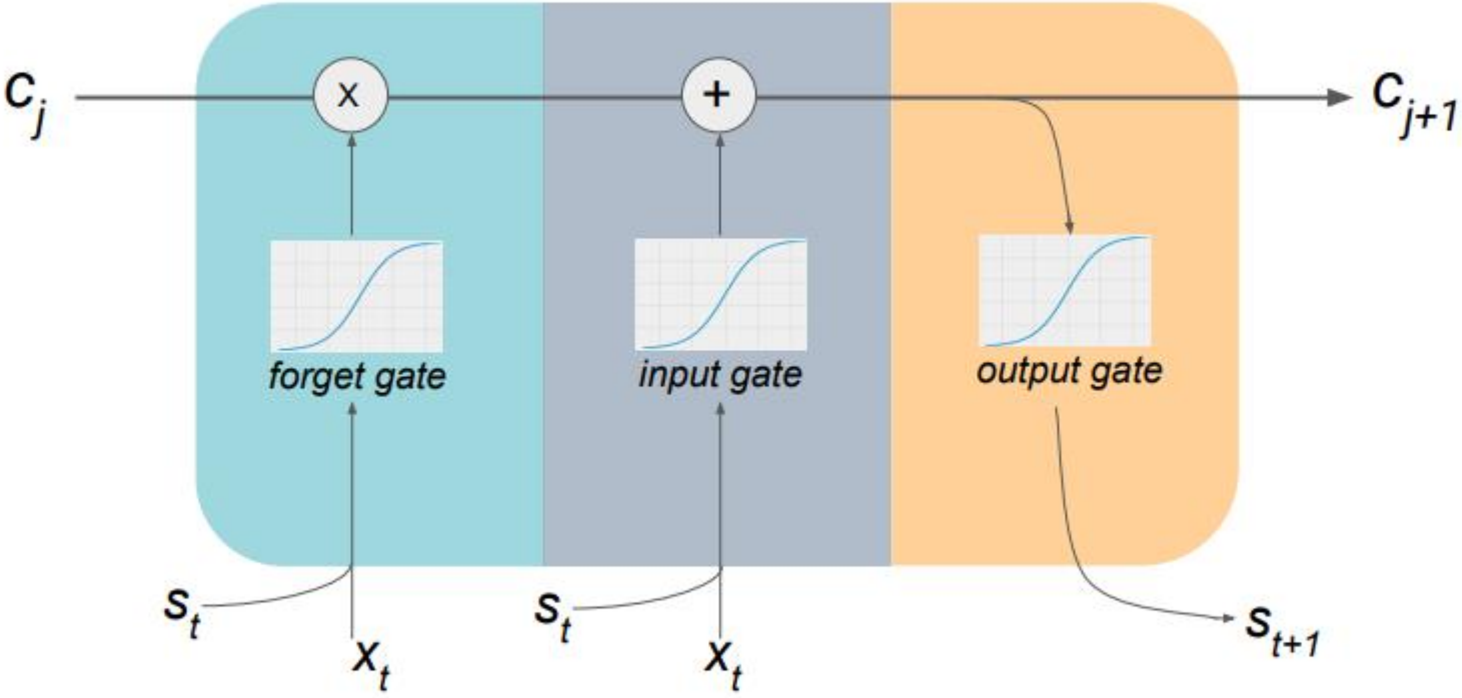
biases initialized to zeros

prevents W from shrinking the gradients

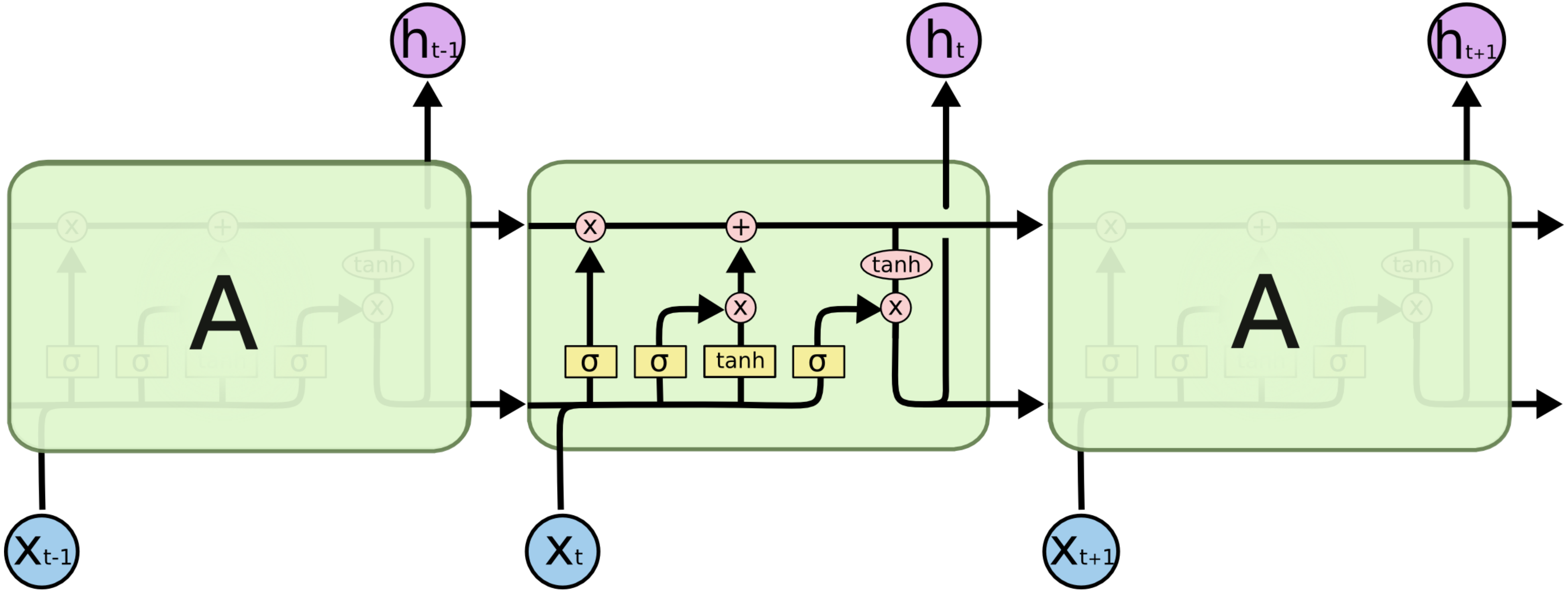
More Robust Solutions: Gated Recurrent Cells



More Robust Solutions: LSTMs

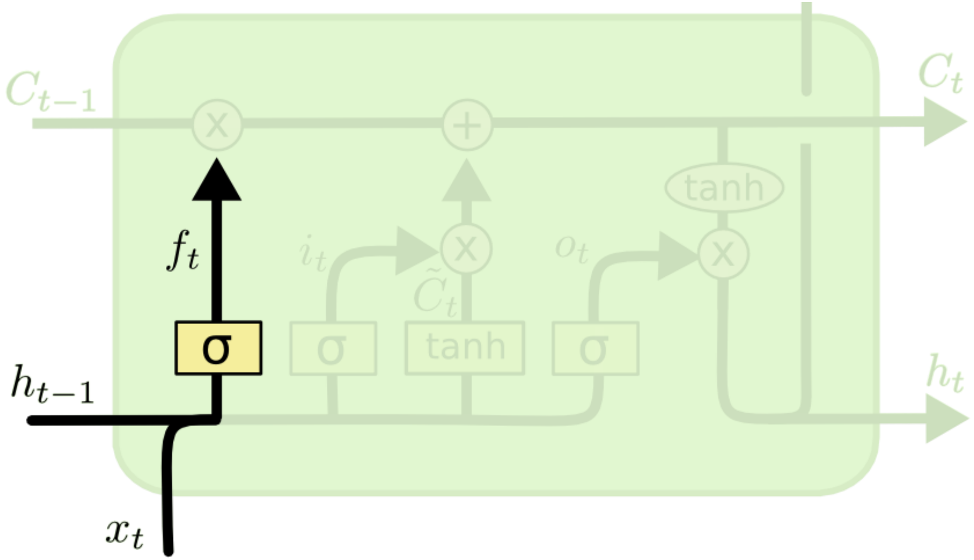


More Robust Solutions: LSTMs



More Robust Solutions: LSTMs

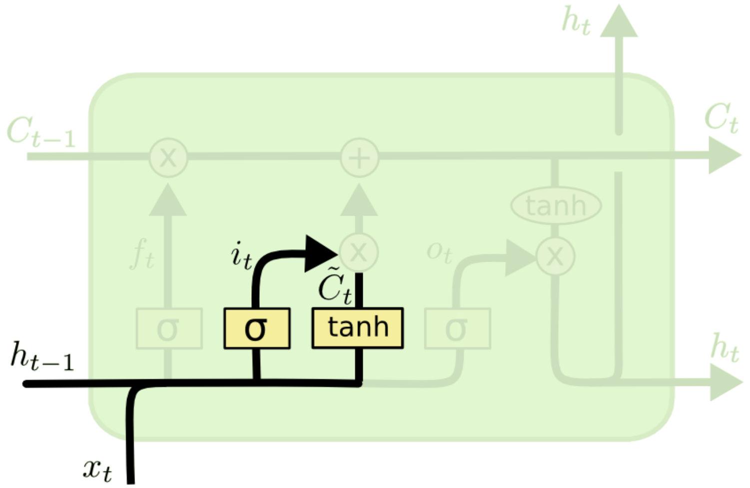
Forget Gate



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

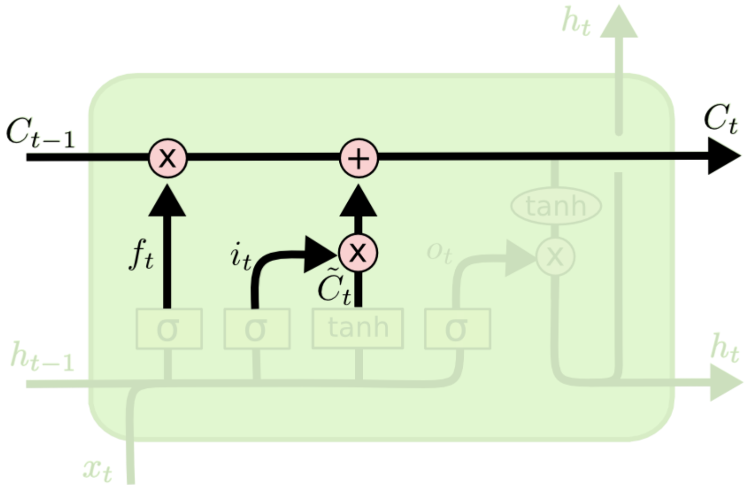
More Robust Solutions: LSTMs

Update Gate



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

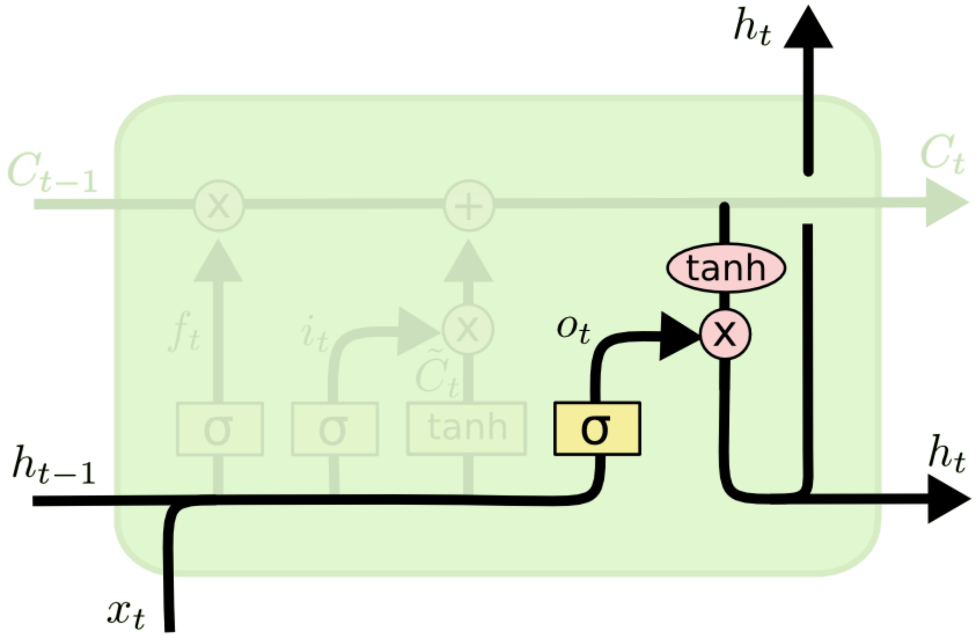
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



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

More Robust Solutions: LSTMs

Output Gate



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

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

Part 2 of RNNs

- **Bidirectional RNNs**
- **Attention**
- **Beam Search**
- **Coding Examples**
  - **Trigger word detection**
  - **Image Captioning(may be)**