

# NEURAL COMPUTATION

Prof. Alexander Huth

4.12.2021

# COURSE ADMIN / PROJECT

- \* Final project planning:
  - \* *Post final project ideas on Canvas discussion (April 12)*
  - \* Respond to someone else's idea (**April 13**)
  - \* Self-organize into groups of 2-4 (groupme)
  - \* Work with group to write proposal & submit via email (**April 14**)

# RECAP

- \* Recurrent neural networks (RNNs)
  - \* Appropriate for problems where data have **sequential dependence**
  - \* Trainable using **backpropagation through time (BPTT)**
    - \* This is like “**unrolling**” the recurrent network into a (very) deep feedforward network, and then applying backpropagation
  - \* RNNs are great for problems like **language**, which requires more sequential processing than vision

# RECAP

- \* Apply the **system construction** approach to the neuroscience of language processing
  - \* Train an RNN **language model** that predicts future words from past
  - \* Use its internal states to predict fMRI data collected while subjects listen to natural language
  - \* This works! But **hierarchy** is difficult to interpret relative to visual CNN models

# THE TROUBLE WITH RNNs

- \* RNNs suffer from the problems of **vanishing** and **exploding gradients**
- \* This problem is worse when backpropagating over a **long** sequence
  - \* Thus limiting the temporal dependence of functions that simple RNNs can learn

# THE TROUBLE WITH RNNs

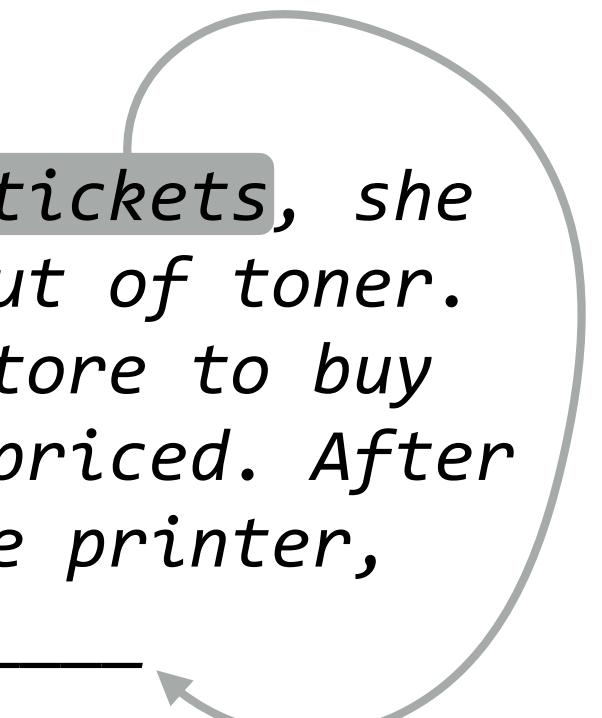
- \* A language model example:

*When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her \_\_\_\_\_*

# THE TROUBLE WITH RNNs

- \* A language model example:

*When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her \_\_\_\_\_*



~37 steps back!

# THE TROUBLE WITH RNNs

- \* Simple 1-D example:

$$h_{t+1} = \sigma(w h_t) \quad \frac{\partial h_{t+1}}{\partial h_t} = w \sigma'(w h_t)$$

# THE TROUBLE WITH RNNs

\* Simple 1-D example:

$$h_{t+1} = \sigma(w h_t) \quad \frac{\partial h_{t+1}}{\partial h_t} = w \sigma'(w h_t)$$

$$\frac{\partial h_{t+2}}{\partial h_t} = w \sigma'(w h_{t+1}) w \sigma'(w h_t)$$

# THE TROUBLE WITH RNNs

\* Simple 1-D example:

$$h_{t+1} = \sigma(w h_t) \quad \frac{\partial h_{t+1}}{\partial h_t} = w \sigma'(w h_t)$$

$$\frac{\partial h_{t+2}}{\partial h_t} = w \sigma'(w h_{t+1}) w \sigma'(w h_t)$$

$$\frac{\partial h_{t+n}}{\partial h_t} = w^n \prod_{j=0}^{n-1} \sigma'(w h_{t+j})$$

# THE TROUBLE WITH RNNs

- \* Simple 1-D example:

$$h_{t+1} = \sigma(w h_t) \quad \frac{\partial h_{t+1}}{\partial h_t} = w \sigma'(w h_t)$$

$$\frac{\partial h_{t+2}}{\partial h_t} = w \sigma'(w h_{t+1}) w \sigma'(w h_t)$$

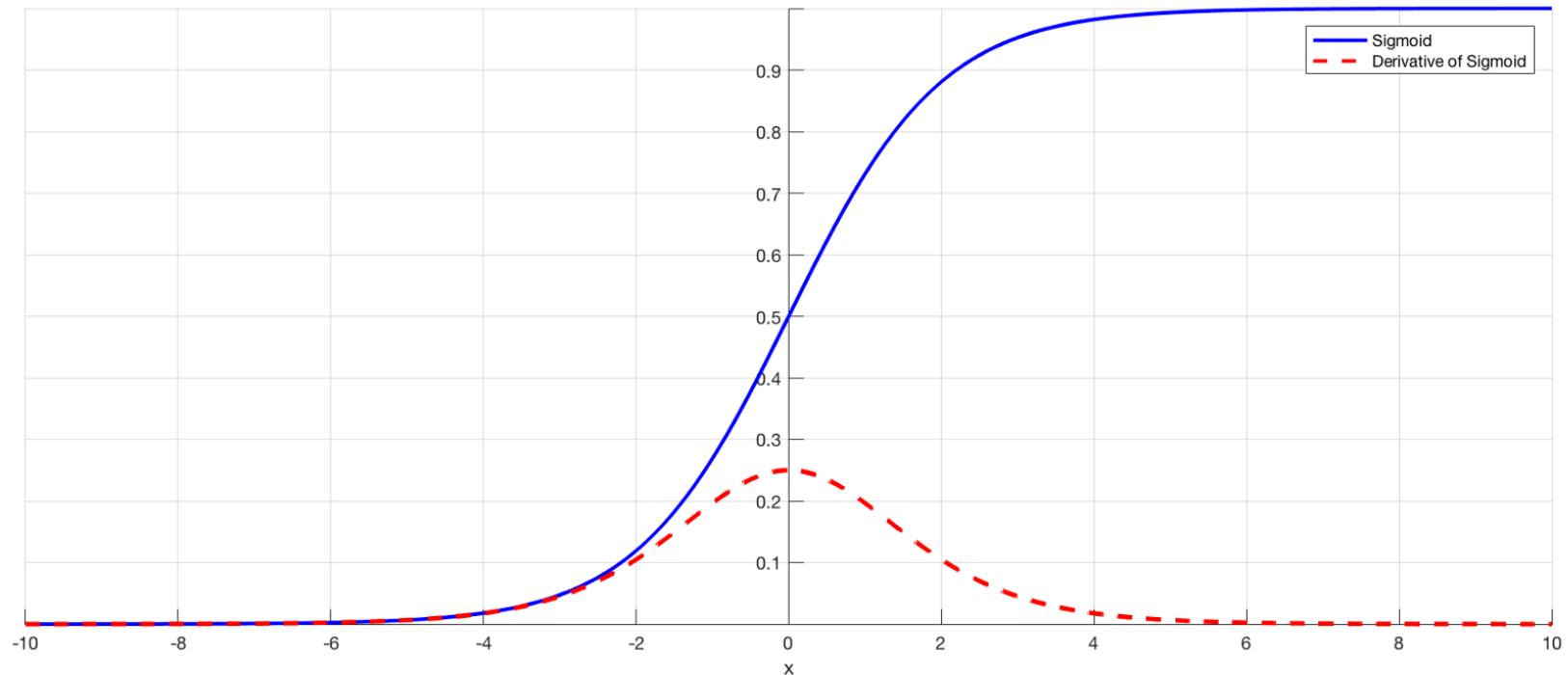
$$\frac{\partial h_{t+n}}{\partial h_t} = w^n \prod_{j=0}^{n-1} \sigma'(w h_{t+j})$$

really bad

also bad

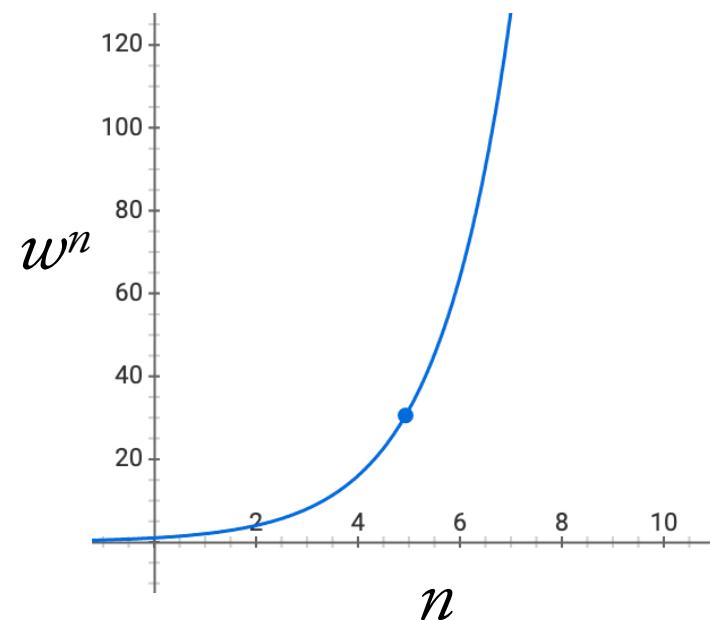
# THE TROUBLE WITH RNNs

- \* **Vanishing gradients** occur if
  - \*  $n$  large,  $|w| < 1$ , or activations  $h_t$  far from 0



# THE TROUBLE WITH RNNs

- \* Exploding gradients occur if
  - \*  $n$  large,  $|w| > 1$



# THE TROUBLE WITH RNNs

- \* How can these things be solved?
  - \* By breaking the chains of (1) weight multiplications, and (2) nonlinearities
- \* Enter: **Long short-term memory networks (LSTMs)**

# LSTM

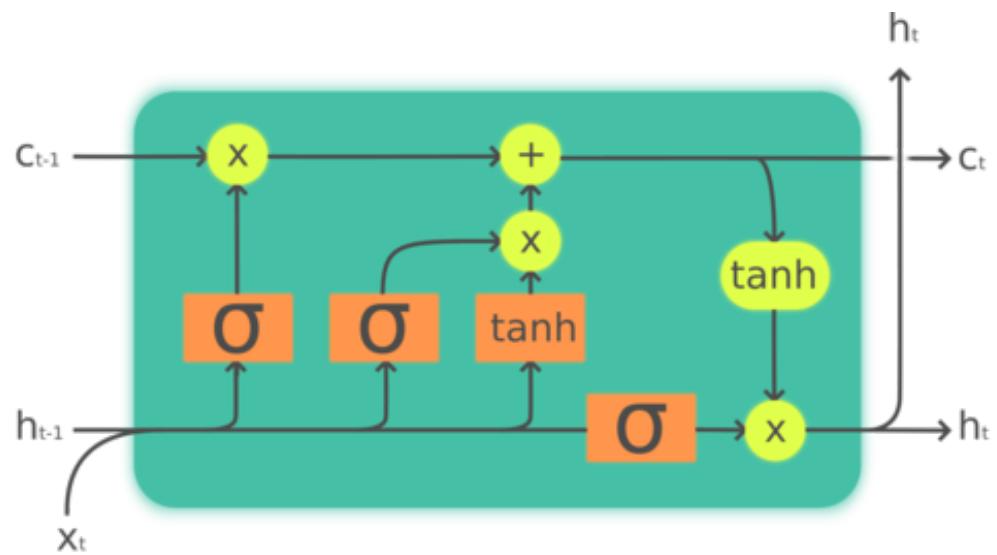


Jürgen Schmidhuber

- \* The LSTM was introduced by Hochreiter & Schmidhuber in 1997
- \* One example of a **gated recurrent network**

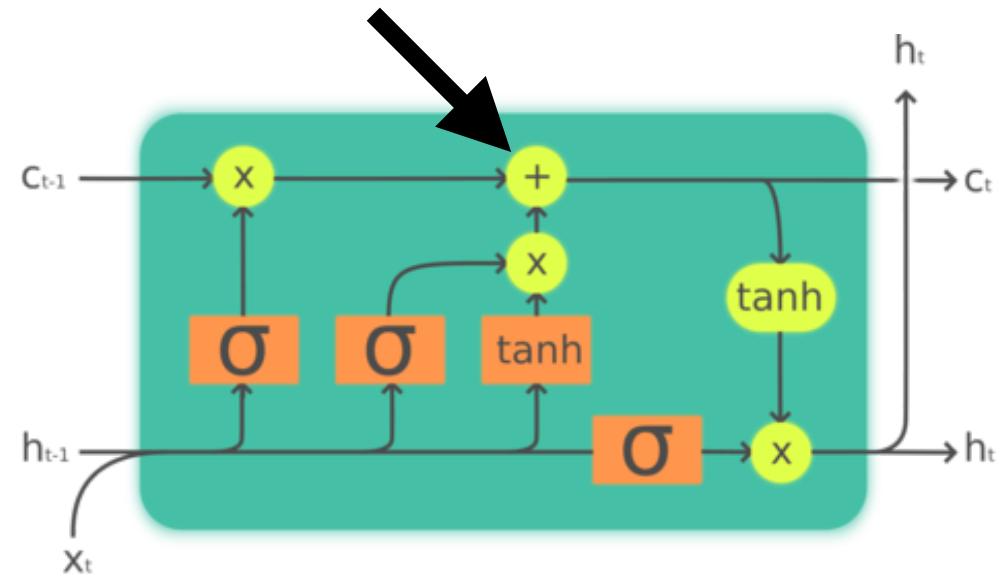
# LSTM

- \* Basic idea:
- \* In addition to the **hidden state**, maintain a separate **cell state** ( $c_t$ ) that does not pass through a non-linearity at each timestep



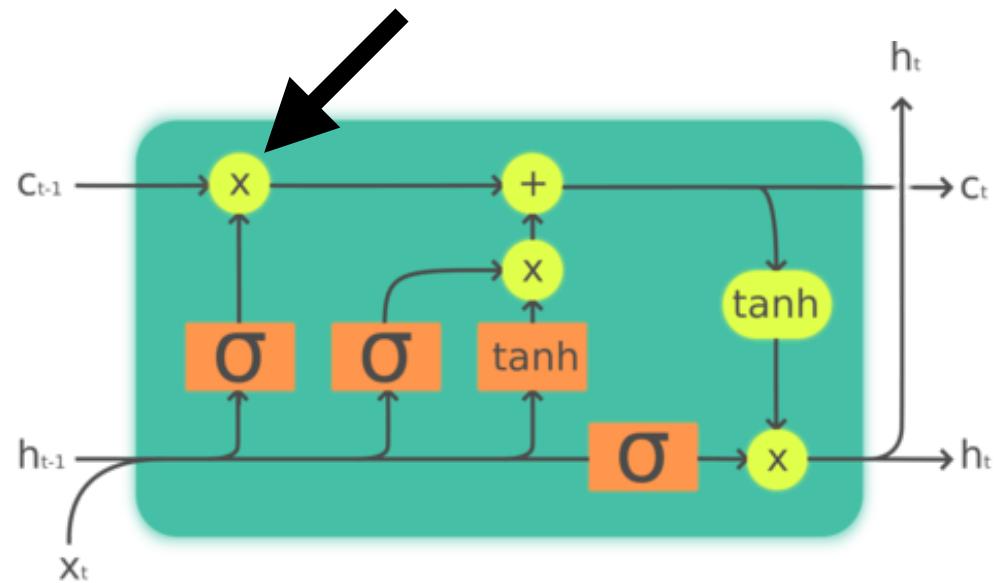
# LSTM

- \* Information can be **added to** the cell state at each timestep



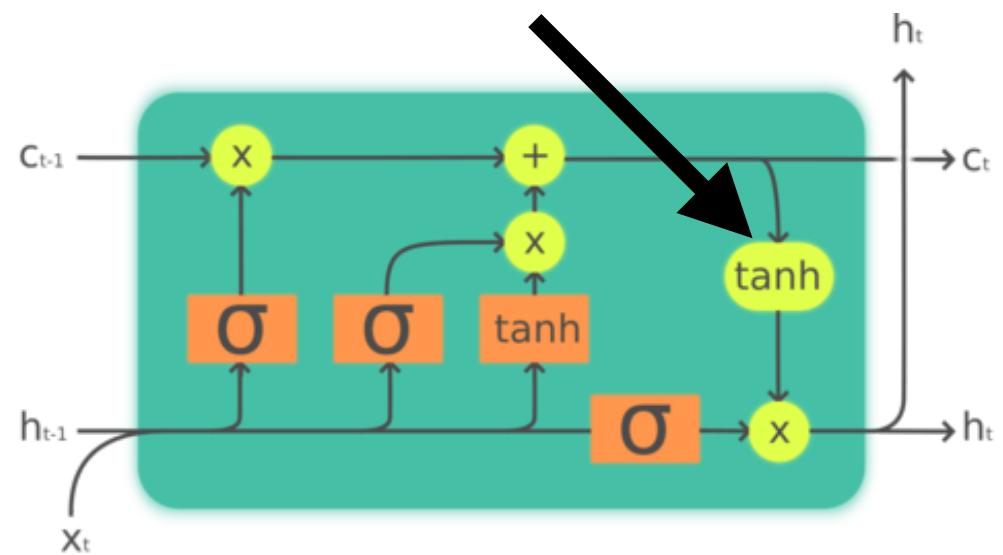
# LSTM

- \* Information can also be **removed from** the cell state (or “forgotten”) at each timestep



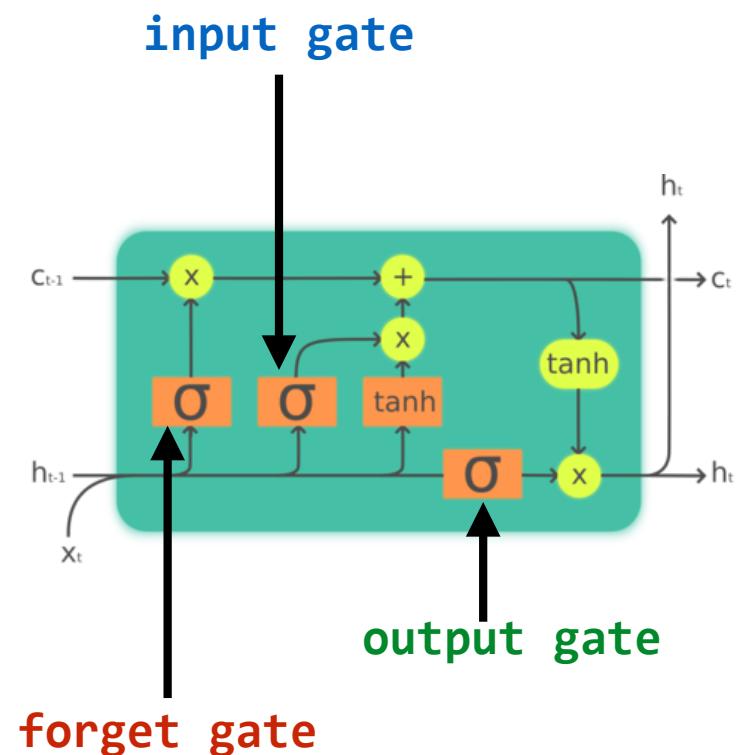
# LSTM

- \* The new cell state is then used to construct the **output** “hidden” state



# LSTM

- \* The LSTM uses a set of **gates** that can open or close to allow information to flow
  - \* The **input gate** controls how information is added to the cell state
  - \* The **forget gate** controls how information is removed from the cell state
  - \* The **output gate** controls how information from the cell state becomes output



# LSTM

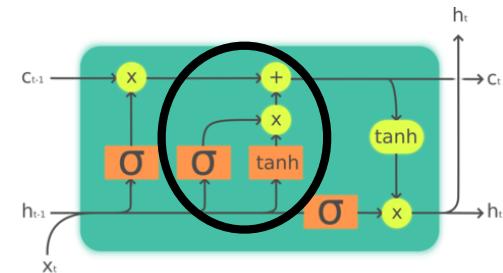
- \* Each gate is computed as a learnable function of:  
**current input** and **previous hidden state**

*Example:  
forget gate*

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

*sigmoid nonlinearity (0...1)*

# LSTM - INPUT GATE



- \* The **input gate** is multiplied by a **proposed cell state** and then added to the **previous cell state**

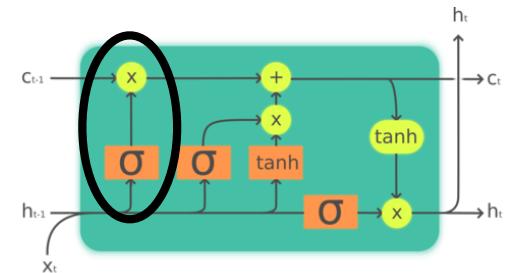
$$\text{input gate: } i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$\text{proposed cell state: } \tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$\text{new cell state: } c_t = \dots + i_t \circ \tilde{c}_t$$

*element-wise or “Hadamard” product*

# LSTM - FORGET GATE

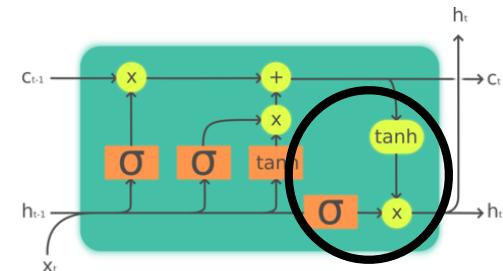


- \* The **forget gate** is multiplied by the **previous cell state** and then added to the **proposed cell state** times the **input gate**

forget gate:  $f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$

new cell state:  $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

# LSTM - OUTPUT GATE

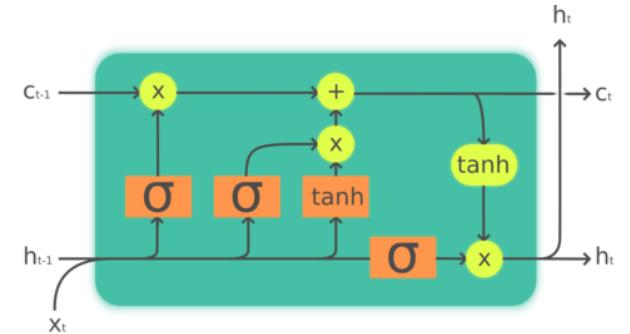


- \* The **output gate** is multiplied by the **current cell state** (with non-linearity applied) to form the **new hidden state**

$$\text{output gate: } o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\text{new hidden state: } h_t = o_t \circ \tanh(c_t)$$

# LSTM



- \* All together now!

- \* Input gate:  $i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$

- \* Forget gate:  $f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$

- \* Output gate:  $o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$

- \* Proposed cell:  $\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$

- \* New cell state:  $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

- \* New hidden state:  $h_t = o_t \circ \tanh(c_t)$

# LSTM

- \* How does the LSTM solve the problems of exploding & vanishing gradients?
- \* Similar to our earlier 1-D example:

$$c_{t+1} = f_{t+1} \circ c_t + \dots \quad \frac{\partial c_{t+1}}{\partial c_t} \approx f_{t+1}$$

# LSTM

- \* How does the LSTM solve the problems of exploding & vanishing gradients?
- \* Similar to our earlier 1-D example:

$$c_{t+1} = f_{t+1} \circ c_t + \dots$$

$$\frac{\partial c_{t+1}}{\partial c_t} \approx f_{t+1}$$

$$\frac{\partial c_{t+n}}{\partial c_t} \approx \prod_{j=1}^n f_{t+j}$$

# LSTM

- \* How does the LSTM solve the problems of exploding & vanishing gradients?
- \* Similar to our earlier 1-D example:

$$c_{t+1} = f_{t+1} \circ c_t + \dots$$

$$\frac{\partial c_{t+1}}{\partial c_t} \approx f_{t+1}$$

*can shrink (a bit)  
but not explode!* →

$$\frac{\partial c_{t+n}}{\partial c_t} \approx \prod_{j=1}^n f_{t+j}$$

# LSTM VISUALIZATION

- \* Suppose we train an LSTM to predicts the **next character** in a sequence from the previous characters
  - \* This network is trained either on text (e.g. *War and Peace*) or C++ code (e.g. the Linux kernel)
- \* Visualizing the **cell state values** (for a few units) at each character can show a bit of what the network has learned!

*From: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>*

# LSTM VISUALIZATION

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

From: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

# LSTM VISUALIZATION

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data))) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

*From: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>*

# LSTMS IN SYSTEM CONSTRUCTION

- \* Jain & Huth (2018) used (word-level) LSTM language models to extract features for modeling fMRI responses to natural language

# NEXT TIME

- \* A different way to get to gated recurrent networks
- \* Interpreting language model representations