LSTM Cells -

Deep Learning for Sequences

By Thomas Klein

tklein@uni-osnabrueck.de

Generating Hype

LSTMs are used in applications like…

1. Google Translate
2. Siri
3. Amazon Alexa
4. AlphaGo
5. …

Anything that deals with **sequential data**

Topics today

1. When can I use LSTMs?

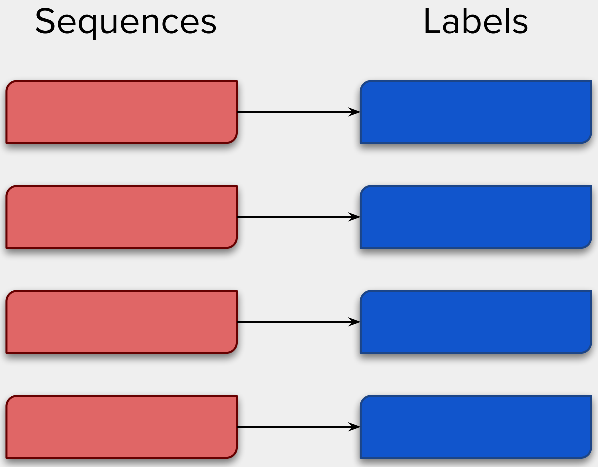
2. What are Artificial Neural Networks again?

3. Neural Nets for sequences = RNNs

4. Finally, LSTMs

High Level Overview

What we have:



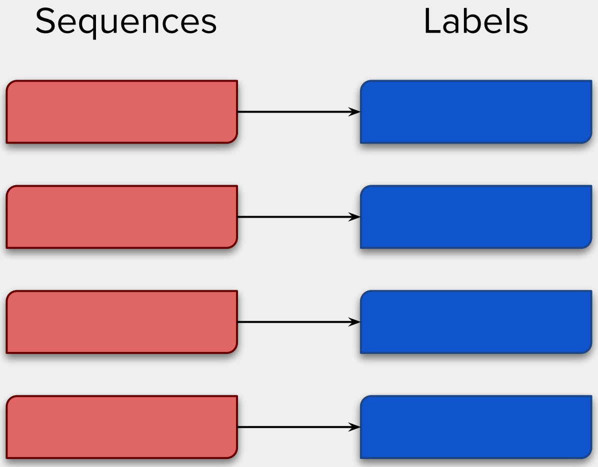
examples for sequences:

- binary: 0011011101101
- text: “Lorem ipsum dolor sit amet”
- sensor measurements over time
- ...

transformed to (high dimensional) real-valued sequences for LSTMs

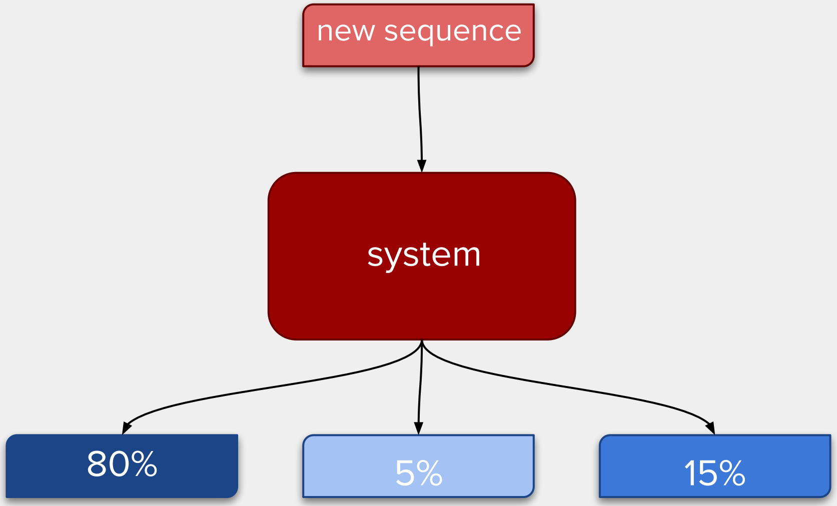
High Level Overview

What we have:



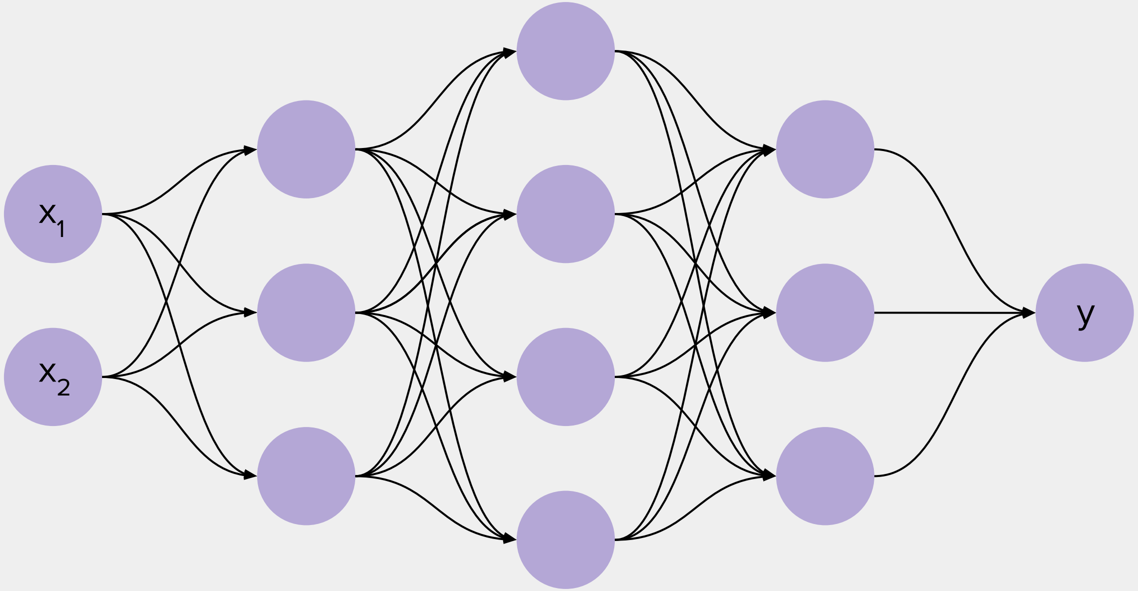
e.g. movie reviews and ratings

What we want:



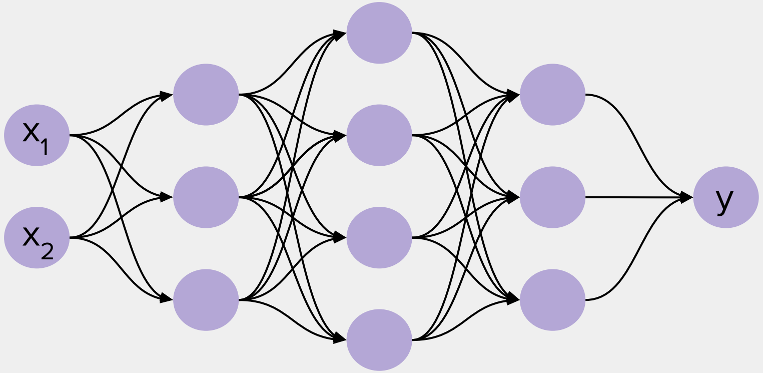
System that learns to infer the correct label from raw data

Artificial Neural Networks



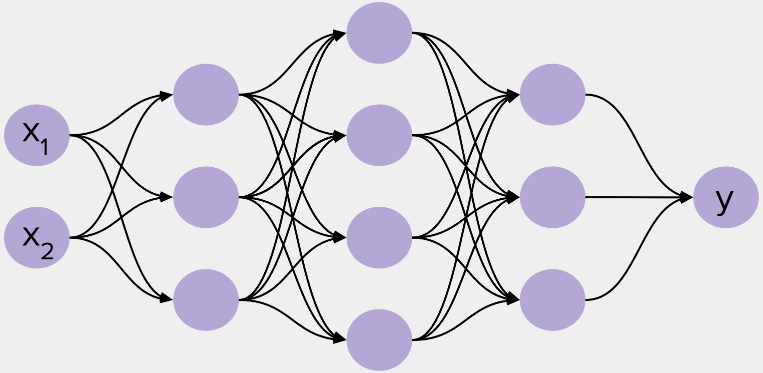
Artificial Neural Networks

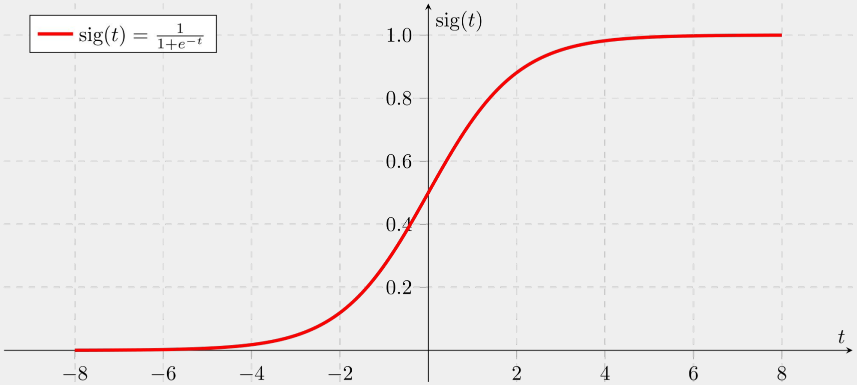
- artificial neuron sums input up and feeds result through activation function



Artificial Neural Networks

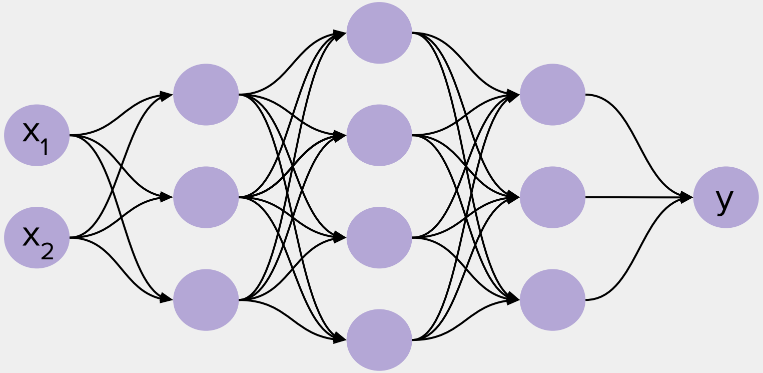
- artificial neuron sums input up and feeds result through activation function

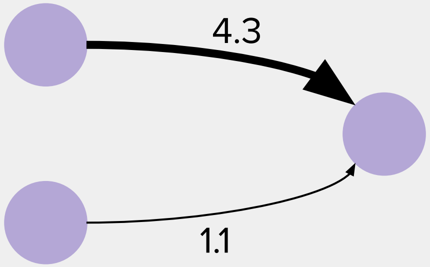




Artificial Neural Networks

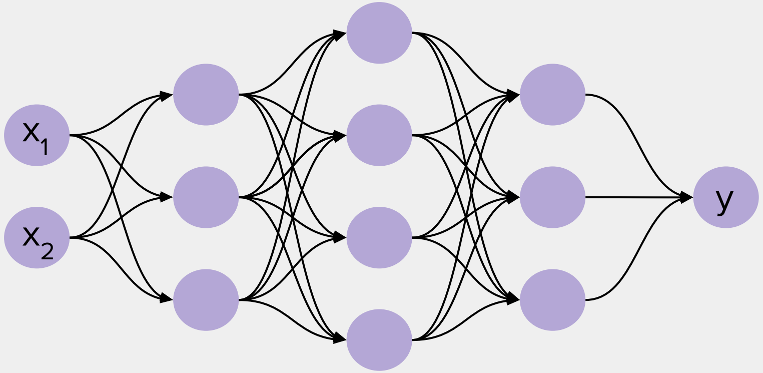
- artificial neuron sums input up and feeds result through activation function
- outputs are weighted by “strength” of connection that they flow through





Artificial Neural Networks

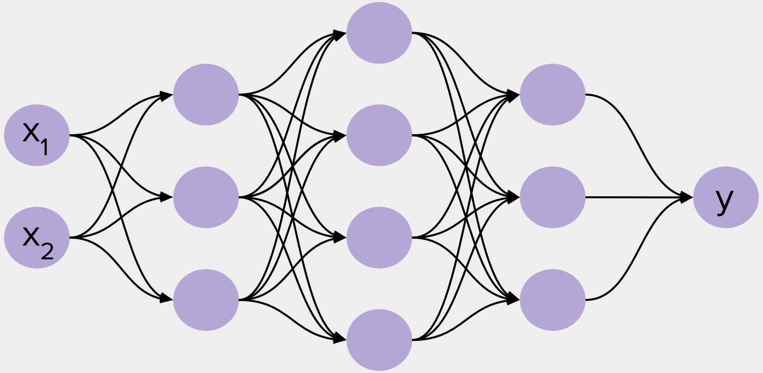
- artificial neuron sums input up and feeds result through activation function
- outputs are weighted by “strength” of connection that they flow through
- those weights are initialized randomly and adapted through an algorithm



Backpropagation

Artificial Neural Networks

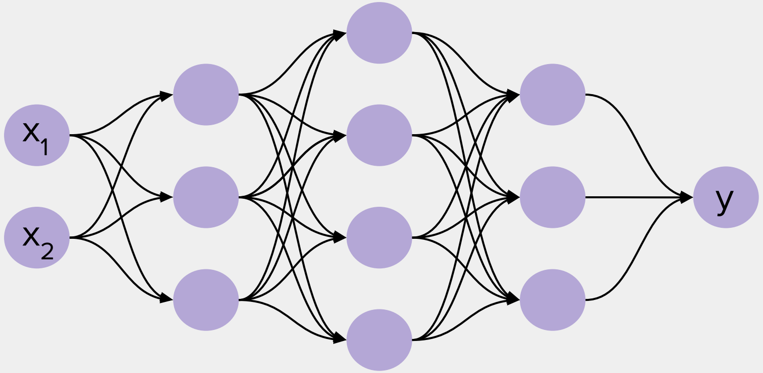
- artificial neuron sums input up and feeds result through activation function
- outputs are weighted by “strength” of connection that they flow through
- those weights are initialized randomly and adapted through an algorithm
- can approximate any continuous function



output of one layer: $\boldsymbol{\sigma}(Wx + b)$

Artificial Neural Networks

- artificial neuron sums input up and feeds result through activation function
- outputs are weighted by “strength” of connection that they flow through
- those weights are initialized randomly and adapted through an algorithm
- can approximate any continuous function

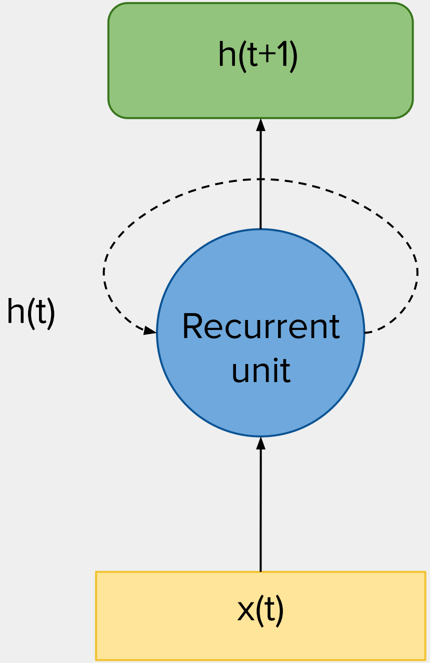


output of one layer: $\boldsymbol{\sigma}(Wx + b)$

… but how that does that work for sequences?

Recurrent Neural Networks

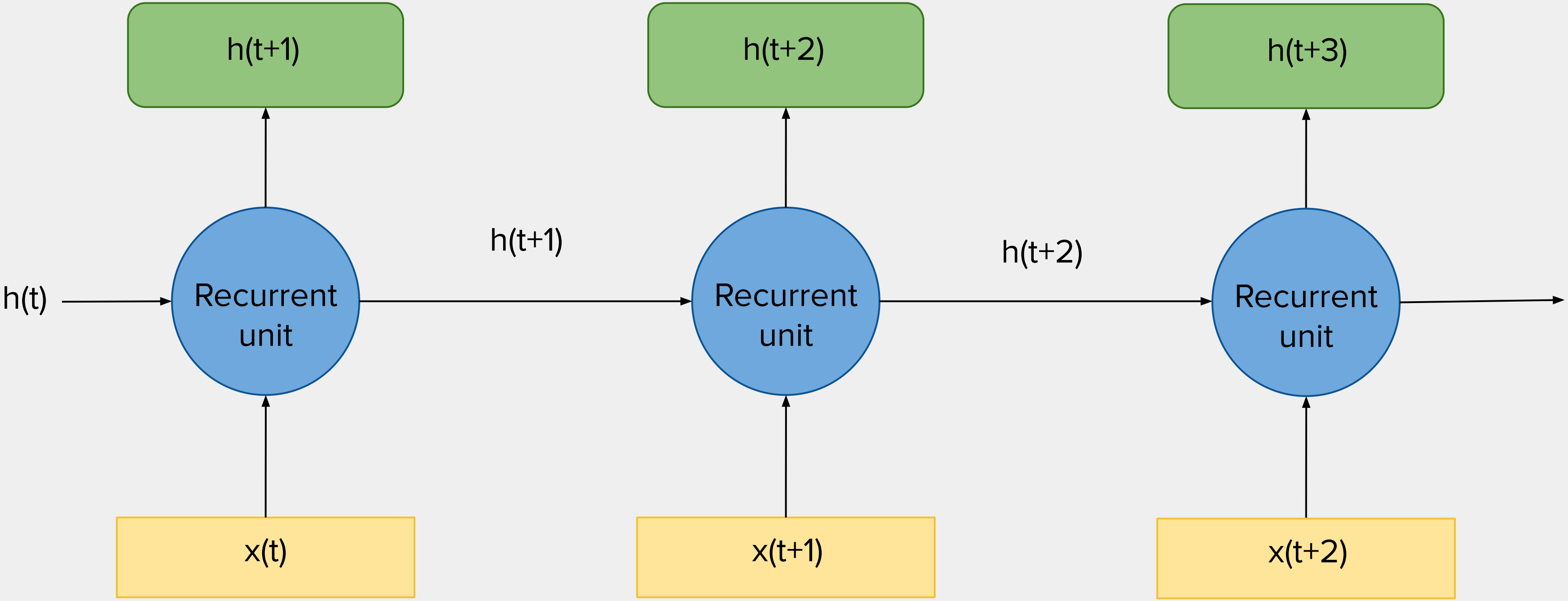
- sequence is fed in step by step
- so, *x(t)* instead of *x*, with *t* = 0, 1, ..., n
- creates immediate output / “hidden state” *h(t+1)*
- neuron reacts not only to *x(t)*, but also to old hidden state *h(t)*
- process the entire sequence, only look at last *h(n)*
- (or look at all *h*’s: sequence-to-sequence)



output of one layer for vanilla RNN:

$$h(t+1) = \sigma(W_{in}x(t) + W_{h}h(t) + b)$$

Unrolling RNNs



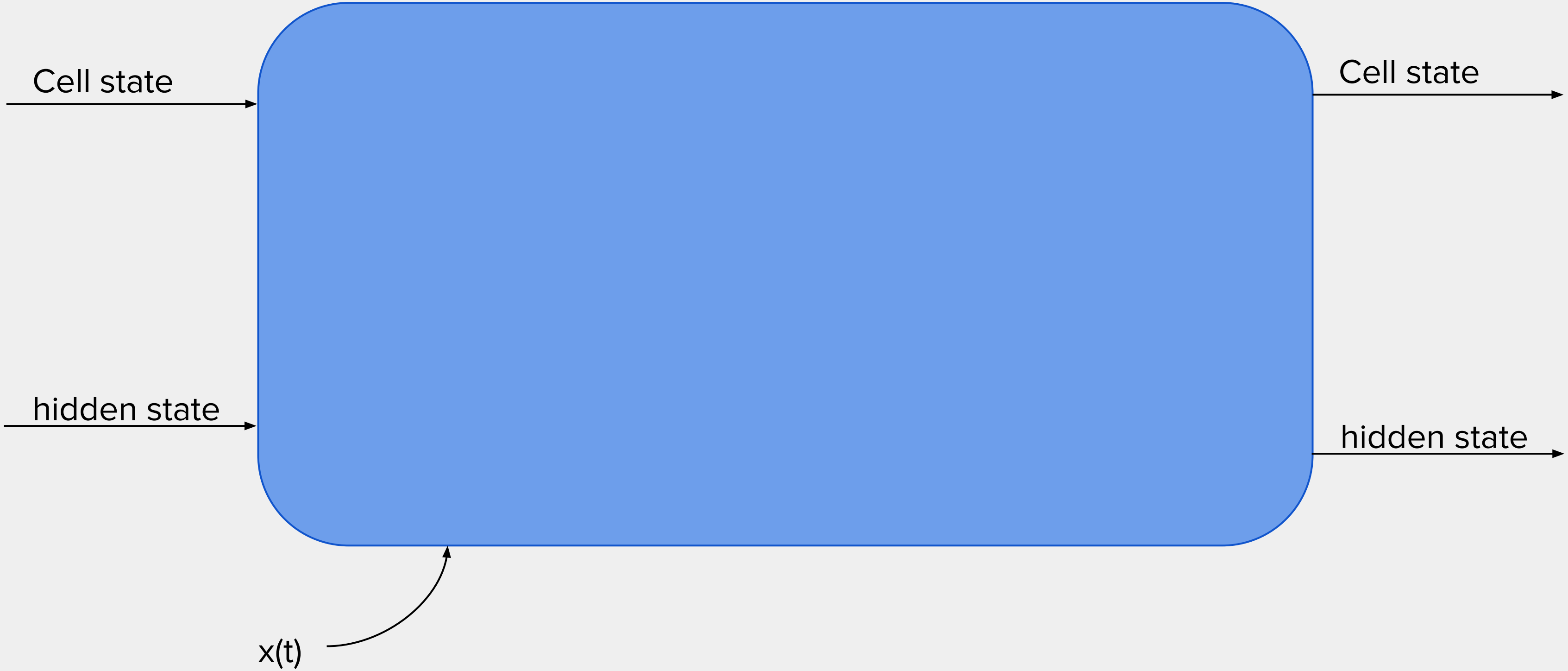
LSTM (finally)

- Long Short-Term Memory, invented by Hochreiter and Schmidhuber in 1997
- special kind of RNN:
  - RNN-problem: “shotgun-memory”, no selection of what to remember
  - example: “I bought a nice pair of sunglasses in France last week.”
  - RNN applies the same weight matrix to every element of the sequence!
  - $h(t+1) = \sigma(W_{\mathrm{in}} x(t) + W_h h(t) + b)$

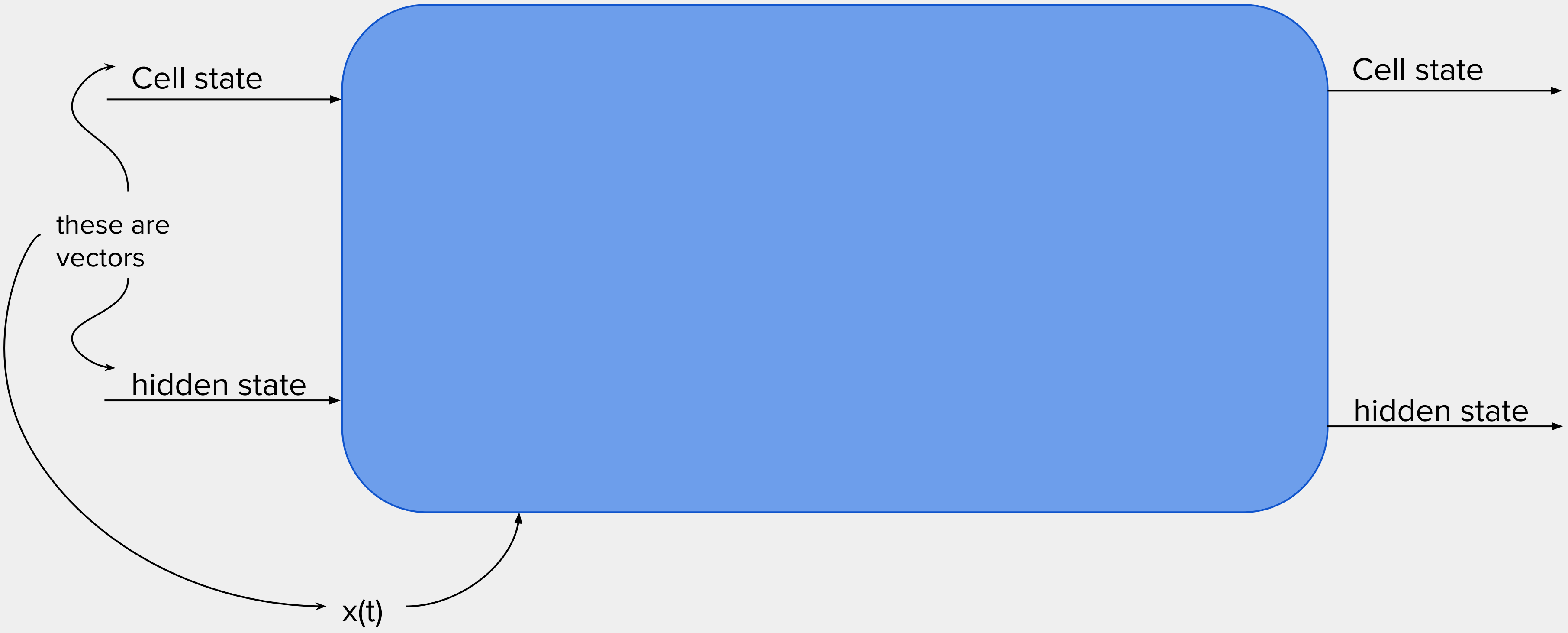
LSTM (finally)

- two kinds of internal states:
  - “hidden state” = short-term memory
  - “cell state” = long-term memory
- update cell state selectively by filtering information through “gates”
  1. forget some old stuff
  2. learn new stuff
  3. output some parts of the new cell state

one step of an unrolled LSTM:



one step of an unrolled LSTM:



Cell state

hidden state

these are vectors

x(t)

Cell state

hidden state

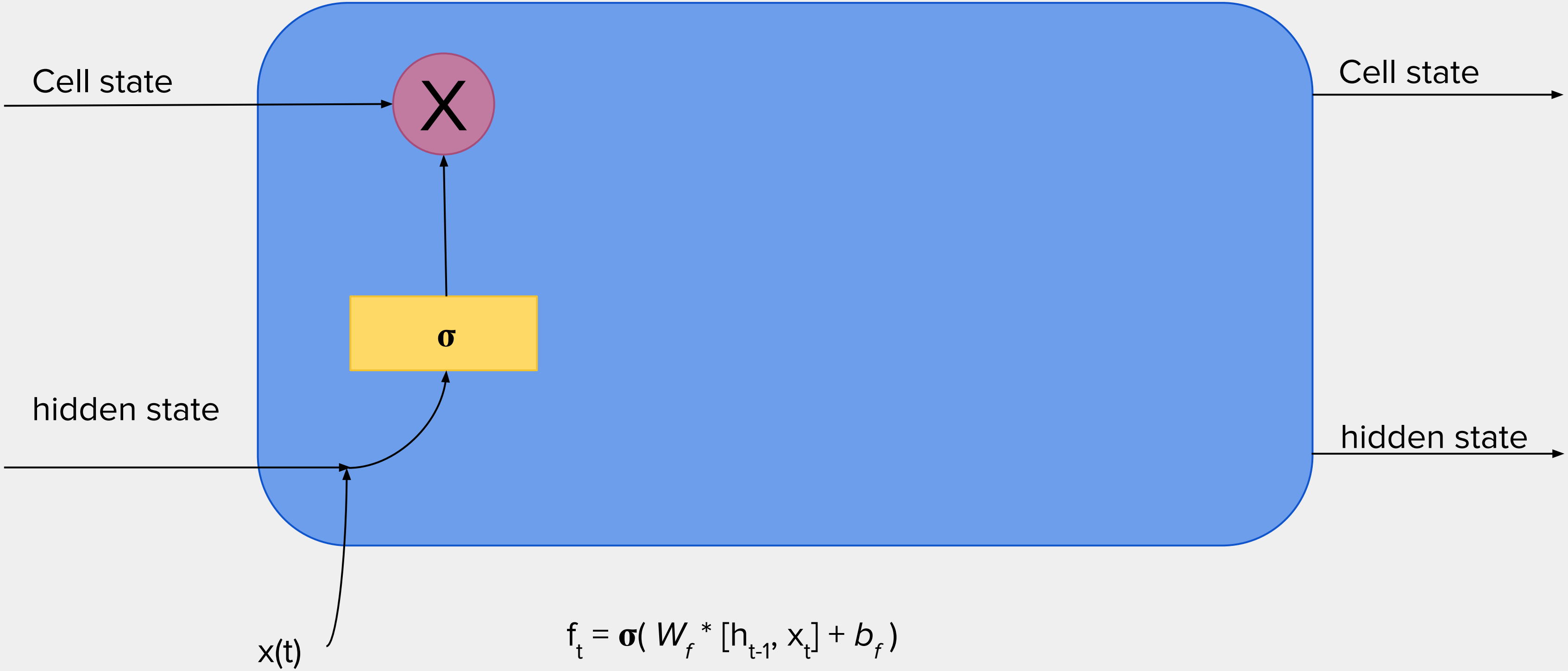
The Three Gates

- Forget Gate
- Input Gate
- Output Gate

The Forget Gate

- we want to filter the cell state
- multiply it element-wise with vector of values between 0 and 1
- obtain those values through a function that depends on hidden state and input
  - learn the function
  - use super simple ANN (one layer, weight-matrix and bias)
  - with sigmoid activation function

1. delete irrelevant input

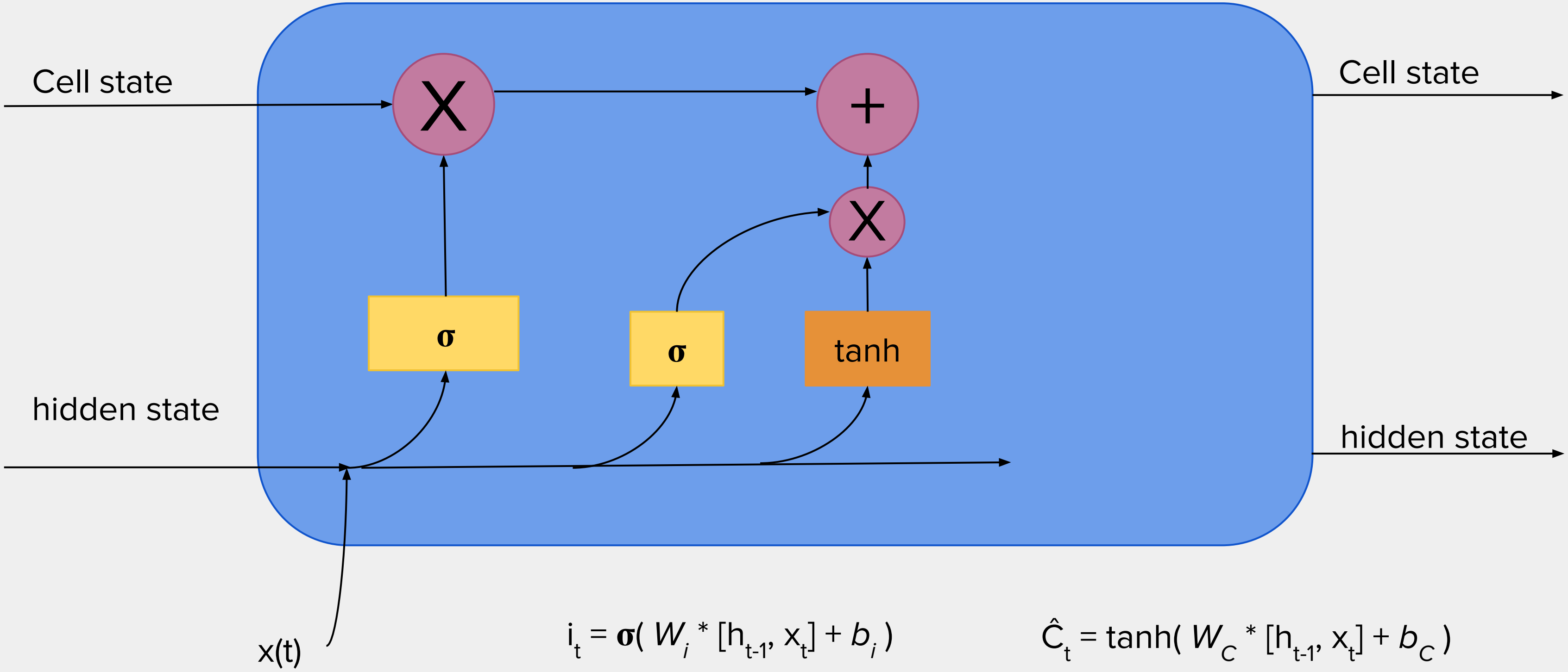


$f_t = \boldsymbol{\sigma}(\, W_f * [h_{t-1}, x_t] + b_f\,)$

The Input Gate

- we want to add new information to the cell state
    1. decide which cells should get information added
    2. add a vector of new values between -1 and 1 to those cells
- select cells just like the forget gate
- use mini-ANN with tanh-activation function to generate new values

2. select and add new input



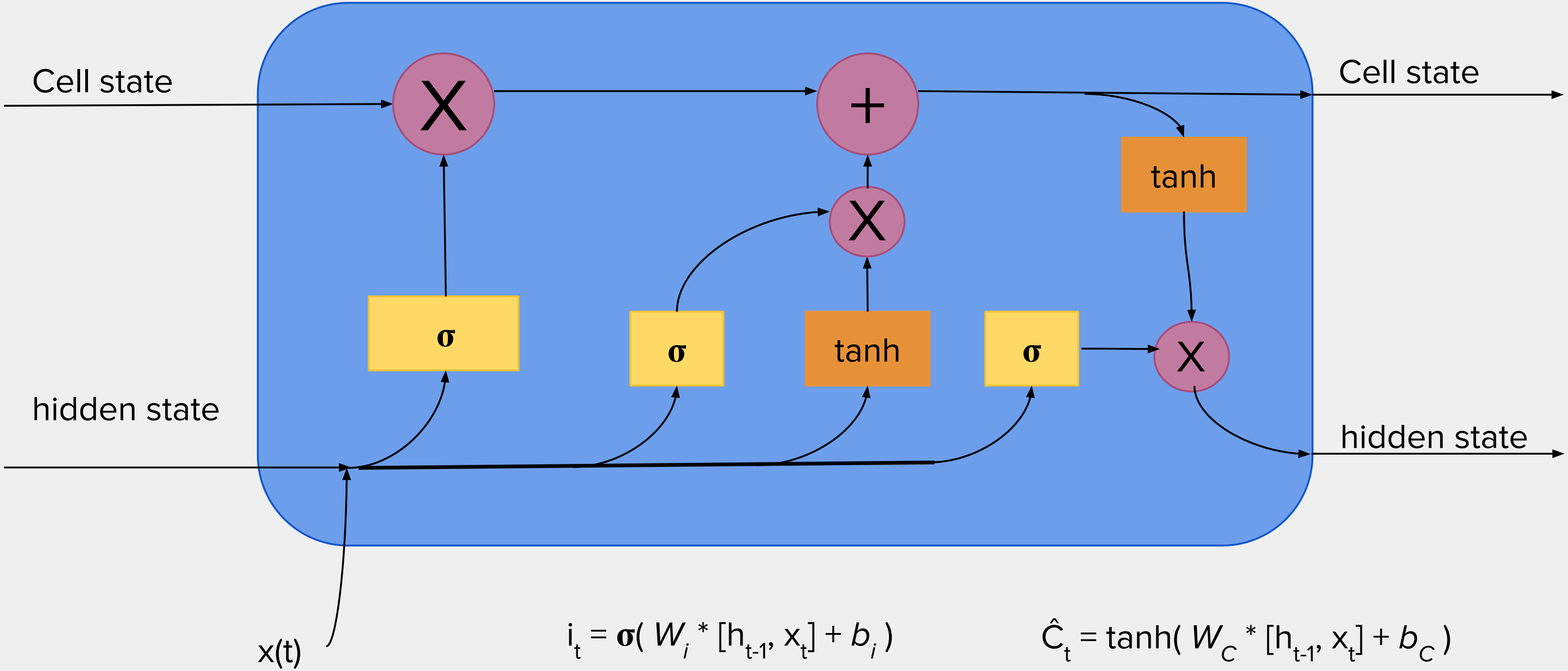
$i_t = \boldsymbol{\sigma}(W_i * [h_{t-1}, x_t] + b_i)$

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

The Output Gate

- we want to select parts of the cell state and output a version of them
    1. decide what we want to output
    2. transform those values
- select cells just like the forget and input gate
- use mini-ANN with tanh-activation function to transform output values

3. return transformed cell state



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

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

Thank you!

(and sorry for the math)

Some excellent resources

- Grant Sanderson (3blue1brown) on ANNs
- Andrej Karpathy’s Blog on RNNs
- Chris Olah’s Blog on LSTMs
- Tensorflow Tutorial for Recurrent Networks
- distill.pub (because not enough people are aware of it)