Friedrich-Alexander-Universität
Technische Fakultät

Machine Learning
Data Analytics

FAU

Machine Learning for Time Series

(MLTS or MLTS-Deluxe Lectures)

Dr. Dario Zanca

Machine Learning and Data Analytics (MaD) Lab
Friedrich-Alexander-Universität Erlangen-Nürnberg
24.01.2023

Topics overview

- Time series fundamentals and definitions (2 lectures)
- Bayesian Inference (1 lecture)
- Gussian processes (2 lectures)
- State space models (2 lectures)
- Autoregressive models (1 lecture)
- Data mining on time series (1 lecture)
- Deep learning on time series (4 lectures) ←
- Domain adaptation (1 lecture)

In this lecture…

1. Long-term dependencies
2. Long-short term memory networks (LSTMs)
3. Other gated architectures

Deep Learning for Time Series – Gated Models

Long-term dependencies

Representation ability of RNNs

RNNs are well suited for tasks involving sequences because they have an internal state that can represent temporal context information.

- The cycles in the graph of an RNN allow to keep information about input where the amount of time (i.e., time steps) is not fixed *a priori*.
- In contrast, feedforward neural networks (FF-NNs) have a “finite impulse response” and can not store information for an indefinite time.
- Parameters of a RNNs can be learn with BPTT, based on a cost function.

Long-term dependencies

Definition. A task is displaying **long-term dependencies** if the prediction of the desired output at time t depends on input presented at an earlier time $\tau \ll t$.

- In presence of long-term dependencies, RNNs can outperform a static FF-NN, although they appear **more difficult to train** optimally.
- Generally, their parameters tend to settle in **sub-optimal solutions that take into account short-term dependences**, but not long-term dependencies.

The symbol „≪” means “much less than”.

Learning long-term dependencies

For a parametric dynamical system that can learn to store relevant state information, we require the following:

1. The system is able to store information for an arbitrary duration (latching information)
2. The system is resistant to noise which is irrelevant to predicting a correct output (robustness).
3. The system‘s parameters are trainable (learnability).

Example: minimal task with long-term dependencies

The following minimal task can be regarded as a test which must be passed in order to satisfy (1), (2), and (3).

Minimal task (for testing a system learning long-term dependencies).

A parametric system is trained to classify two different sets of sequences of an arbitrary lenght T. For each sequence $s = (s_1, \dots, s_T)$, the corresponding class only depends on the initial L values, i.e.,

$$C(s_1, \dots, s_L, \dots, s_T) \equiv C(s_1, \dots, s_L)$$

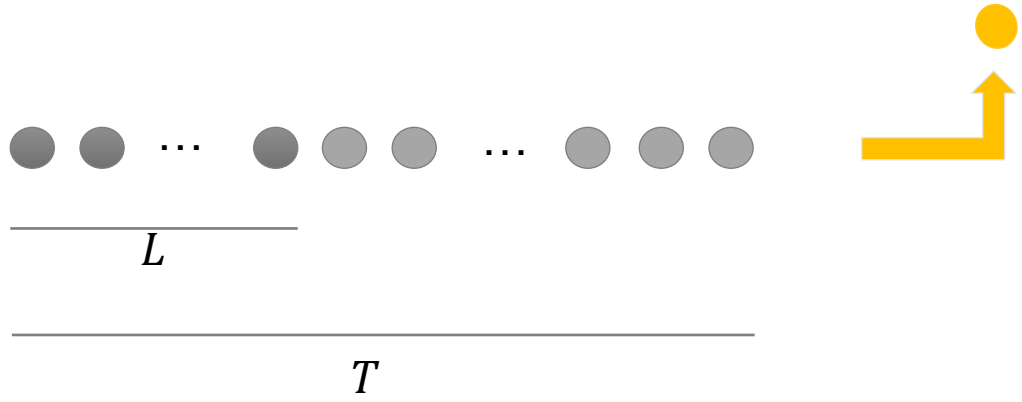
where we suppose L is fixed and $L \ll T$. The system should provide a prediction at the end of each sequence.

"Learning long-term dependencies with gradient descent is difficult", Bengio et al. (1994)

Example: minimal task with long-term dependencies

The problem can be solved only if the system is capable of storing the information about the **initial input values** for an arbitrary duration.

The values $(s_{L+1}, \dots, s_T)$ are irrelevant and can be regarded as *noise*. They can have the effect of erasing the internal information about the initial values of the input.



Example: minimal task with long-term dependencies

The test system has to process one input h_t and one state x_t for each discrete time step.

- The initial inputs h_t with $t \leq L$ contain the relevant information.
- We assume that h_t with $t > L$ is Gaussian noise.
- The connection weights of the test system are trainable.
- Optimization is based on the cost function

$$\mathcal{L} = \frac{1}{2} \sum_{p} \left(x_T^p - d^p \right)^2$$

where $d^p \in \{-1, 1\}$ represents the corresponding class of the input sequence.

Example: minimal task with long-term dependencies

A simple recurrent network candidate solution for the minimal taks is made of a single recurrent neuron.

$$x_t^k = f\left(a_t^k\right) = \tanh\left(a_t^k\right)$$

$$a_t^k = \mathrm{w}\,\mathrm{f}\left(\mathrm{a}_{\mathrm{t}-1}^k\right) + h_t^k$$

$$a_0^0 = a_0^1 = 0$$

where k represents the corresponding sequence class.

It can be shown that, under mild conditions, the dynamic of this neuron system has two attractors and can robustly latch one bit of information represented by the sign of its activation.

Learning long-term dependecies is difficult

In order to latch a bit of information, the system must be able to restrict its activation to a subset S of its domain.

→ In this way it is possible to interpret the activation in two ways, as $a_t \in S$ or $a_t \notin S$.

To make sure the system remains in such a region, the system must be chosen in a way that the region is a basin of attraction of an hyperbolic attractor.

Two conditions can arise when using hyperbolic attractors to latch bits of infomations:

- Either the system is very sensitive to noise,
- Or the derivatives of the cost function at time t with respect to the system activation a_0 converge exponentially to 0.

Vanishing/Exploding gradient problem

Exploding gradient. When the gradients of the loss function with respect to the network parameters are very large.

→ Learning is unstable

→ A possible solution is **gradient clipping,** i.e., cutting off gradients which are greater than a threshold during backpropagation.

Vanishing gradient. When the gradients of the loss function with respect to the network parameters are very small.

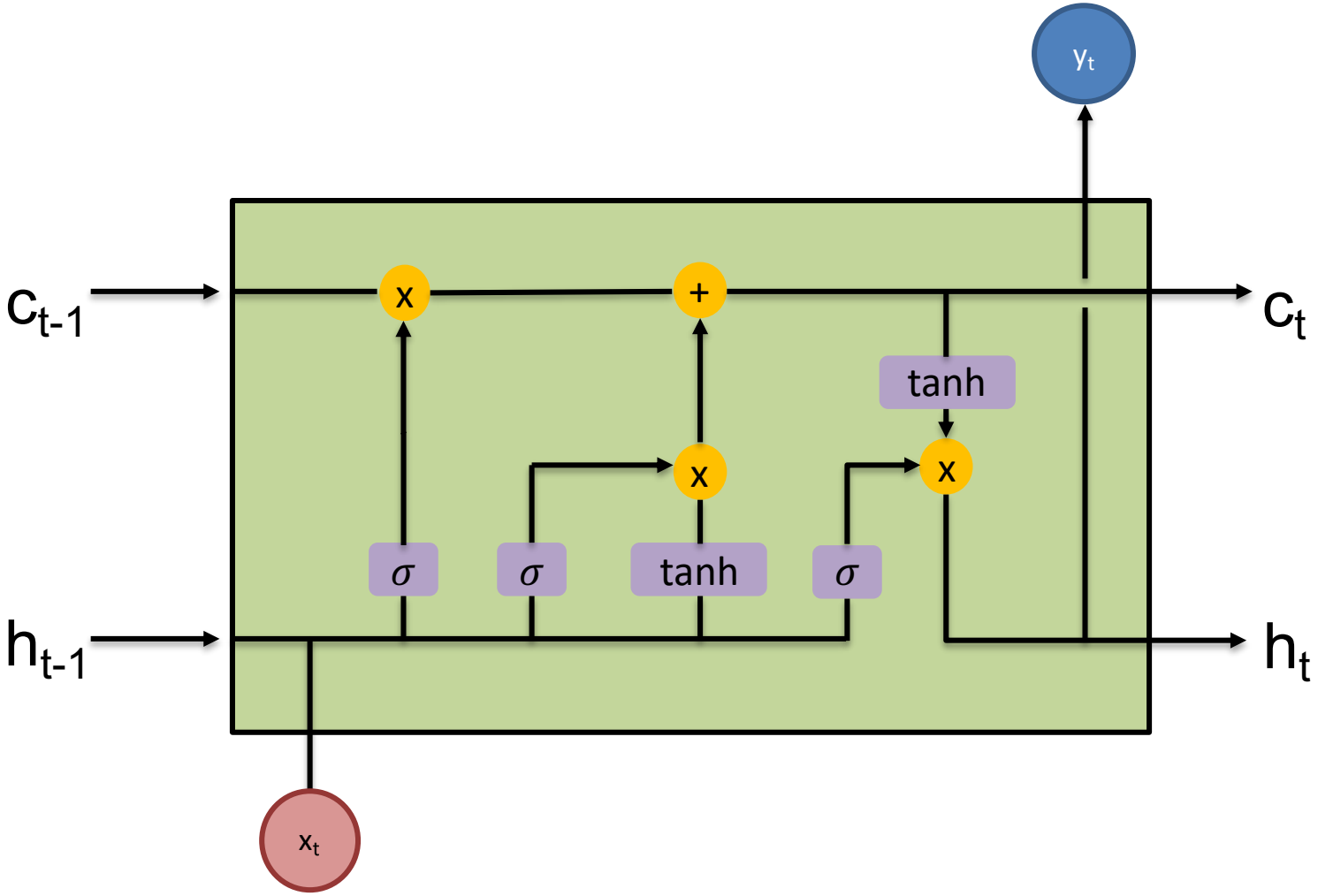
→ Learning is slow

→ A possible solution is to introduce „shortcuts“ in the architecture, e.g., **gates**.

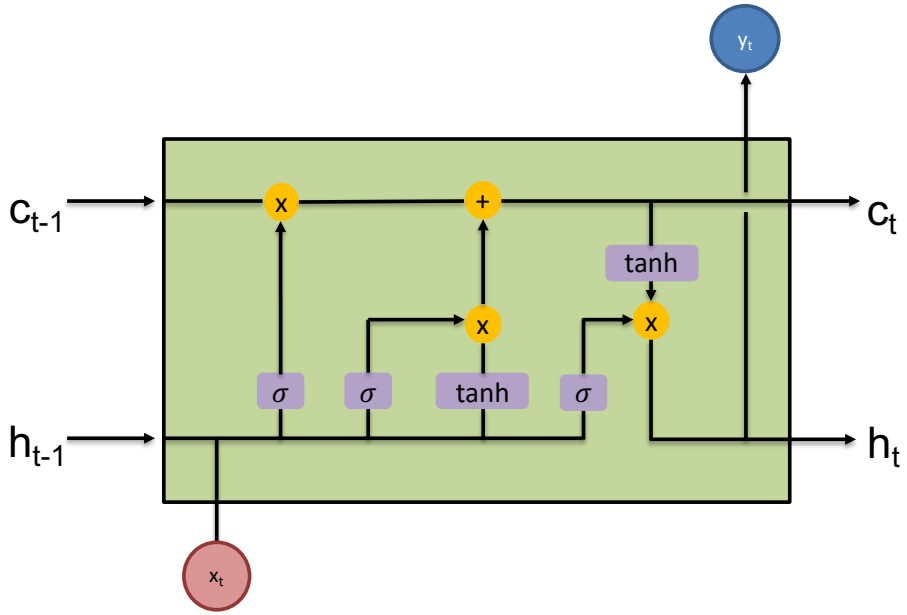
Deep Learning for Time Series – Gated Models

Long-short term memory networks (LSTMs)

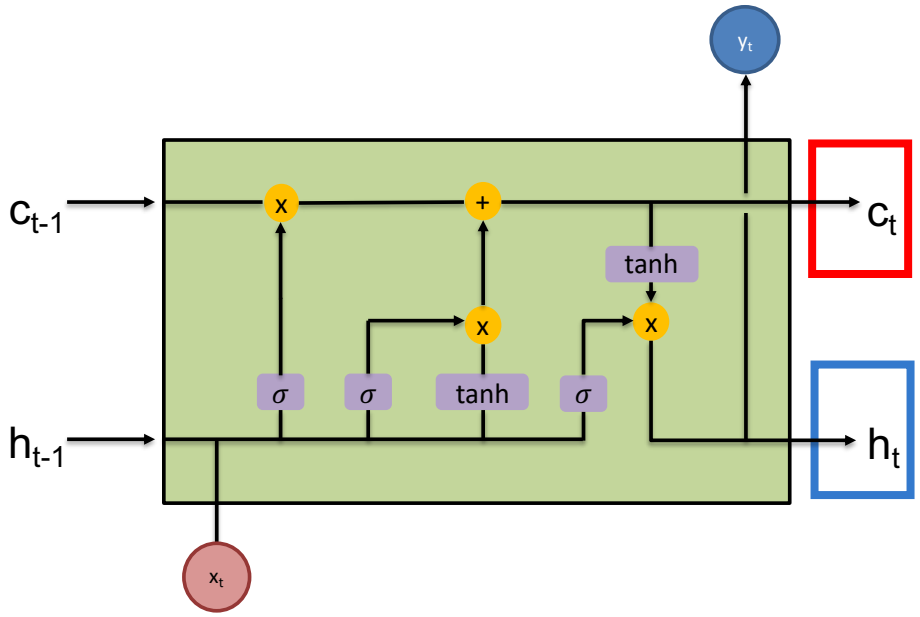
Long-short term memory networks (LSTMs)



Long-short term memory networks (LSTMs)



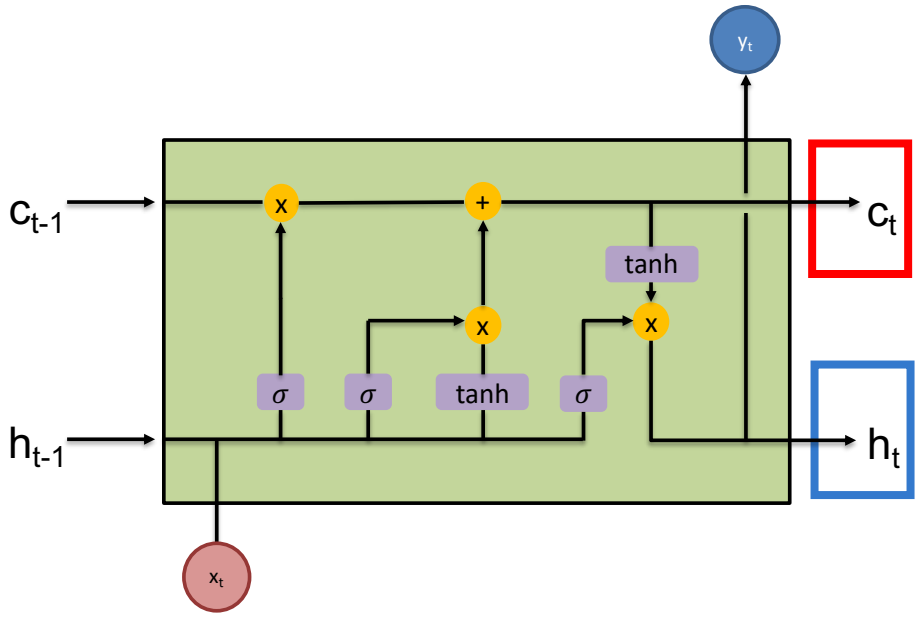
Long-short term memory networks (LSTMs)



Observations:

- Compared to RNNs, additionally to the **hidden state** h_t, we have another state, c_t, said the **cell state**.

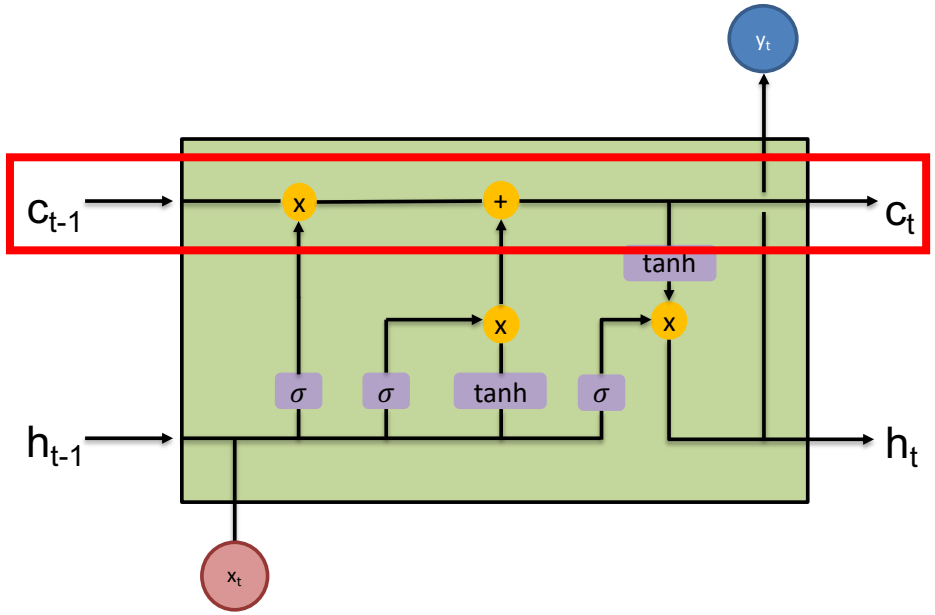
Long-short term memory networks (LSTMs)



Observations:

- Compared to RNNs, additionally to the **hidden state** h_t, we have another state, c_t, said the **cell state**.
- The information flow is regulated by **three gates**: the forget gate, the input gate and the output gate.

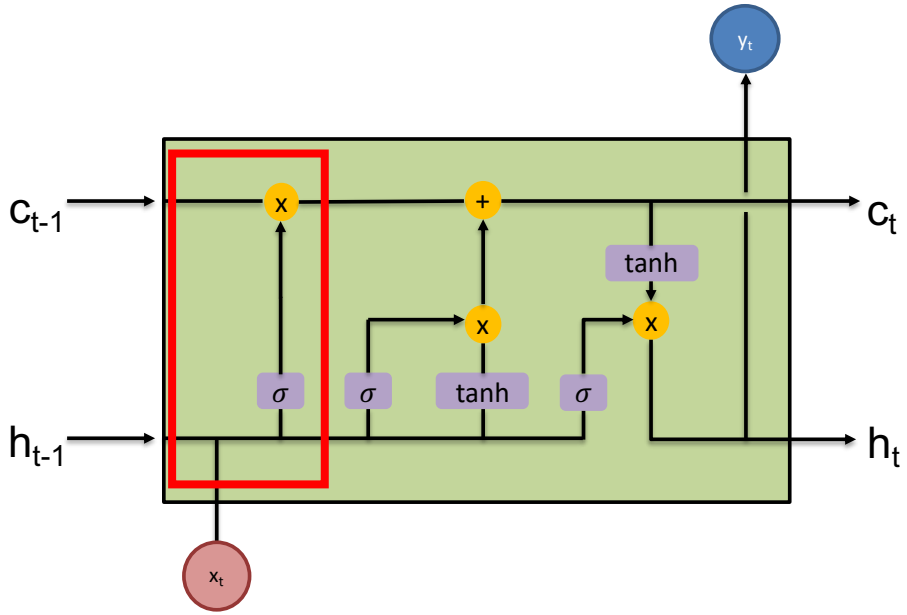
Long-short term memory networks (LSTMs)



The cell state has:

- Only minor interactions
- Simple information flow
- Other gates regulates weather it is preserved/not-preserved or updated.

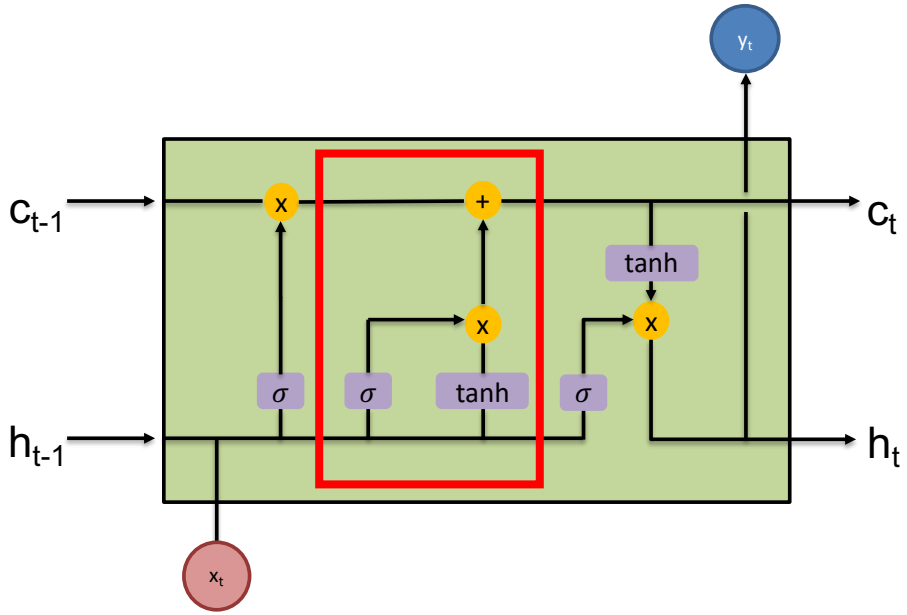
Long-short term memory networks (LSTMs)



The **forget gate** decides how much information to retain from the previous cell state.

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

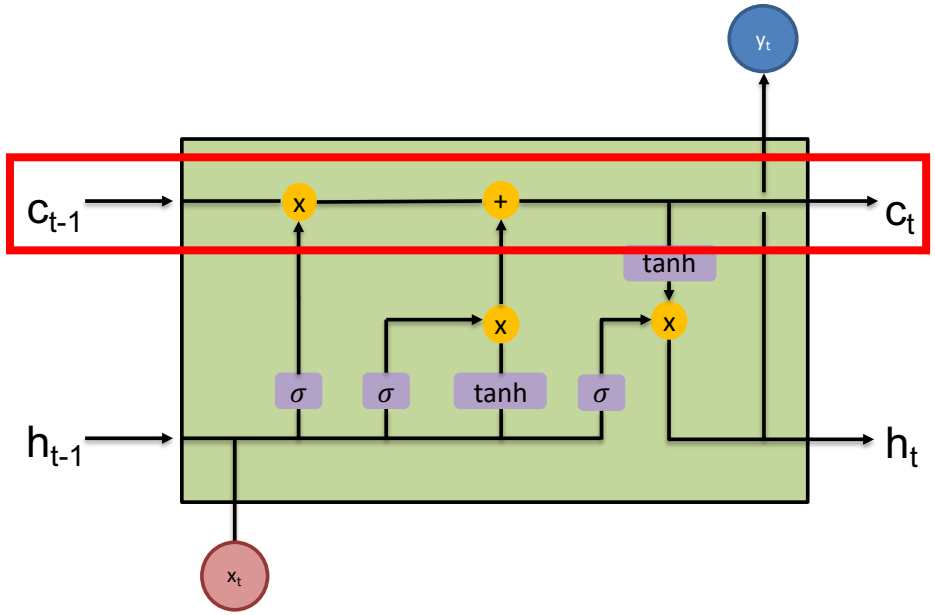
Long-short term memory networks (LSTMs)



The **input gate** decides the information to be added to the cell state, based on the current input.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\mathrm{g_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

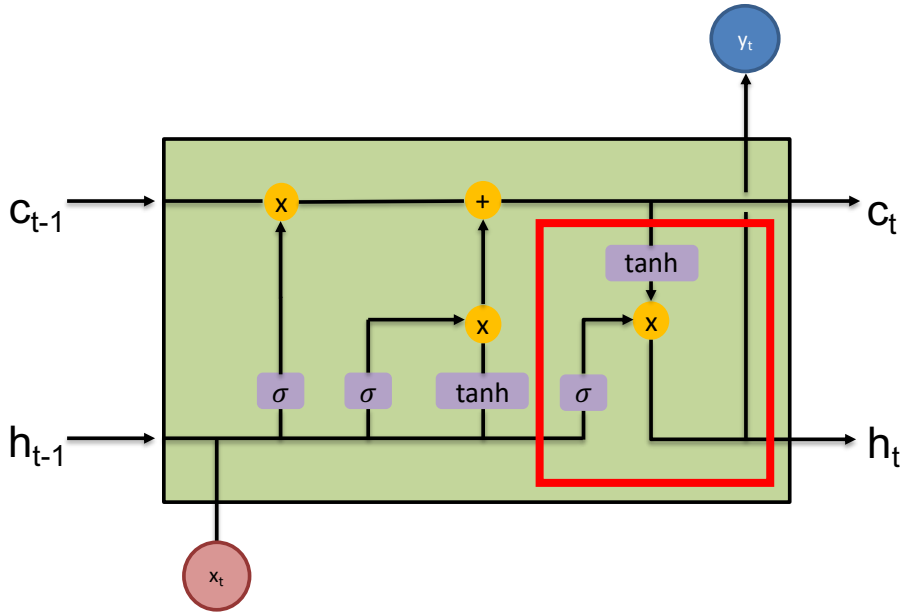
Long-short term memory networks (LSTMs)



The values can be combined to update the cell state

$$c_t = f_t \odot c_{t-1} + i_t \odot \mathrm{g_t}$$

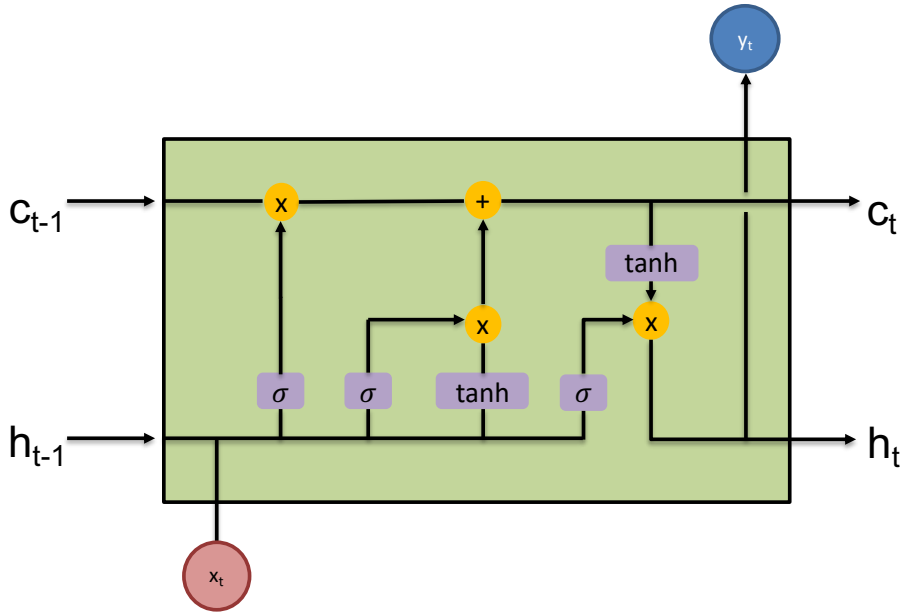
Long-short term memory networks (LSTMs)



The **output gate** generates the output of the current LSTM cell, based on the current input, the previous output, and the updated cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(c_t)$$

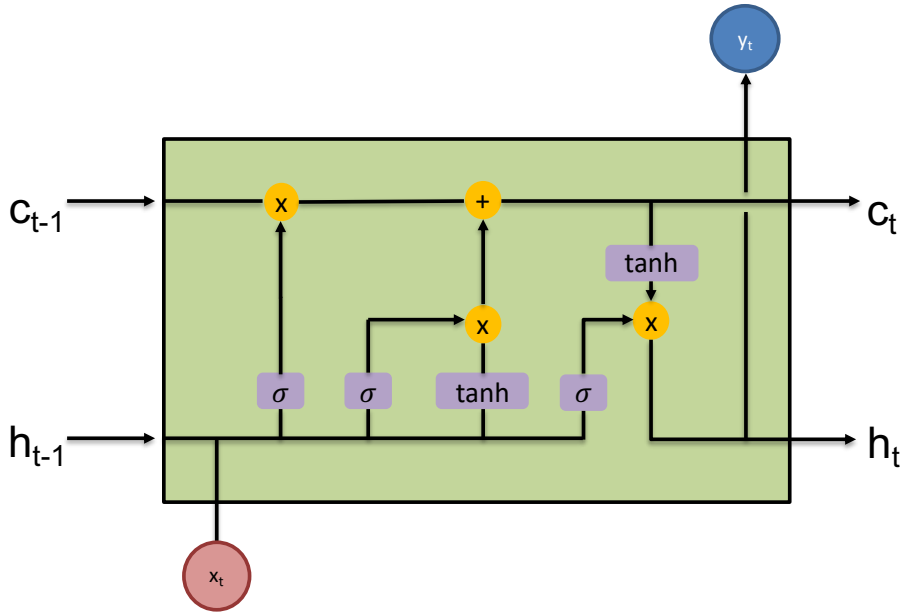
Long-short term memory networks (LSTMs)



To summarize, the LSTM cell is described by the following equations:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\mathrm{g_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \mathrm{g_t}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(c_t)$$

Long-short term memory networks (LSTMs)



From the original publication [1]:

"*Each memory cell's internal architecture guarantees constant error flow within its constant error carrousel CEC… This represents the basis for bridging very long time lags. Two gate units learn to open and close access to error flow within each memory cell's CEC. The multiplicative input gate affords protection of the CEC from perturbation by irrelevant inputs. Likewise, the multiplicative output gate protects other units from perturbation by currently irrelevant memory contents.*"

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Deep Learning for Time Series – Gated Models

Other gated architectures

Motivation

We can define different gated recurrent neural network architectures, e.g.,

→ Fit a specific problem

→ Reduce computations

→ Adapt to different data characteristics

Alternatives to LSTMs:

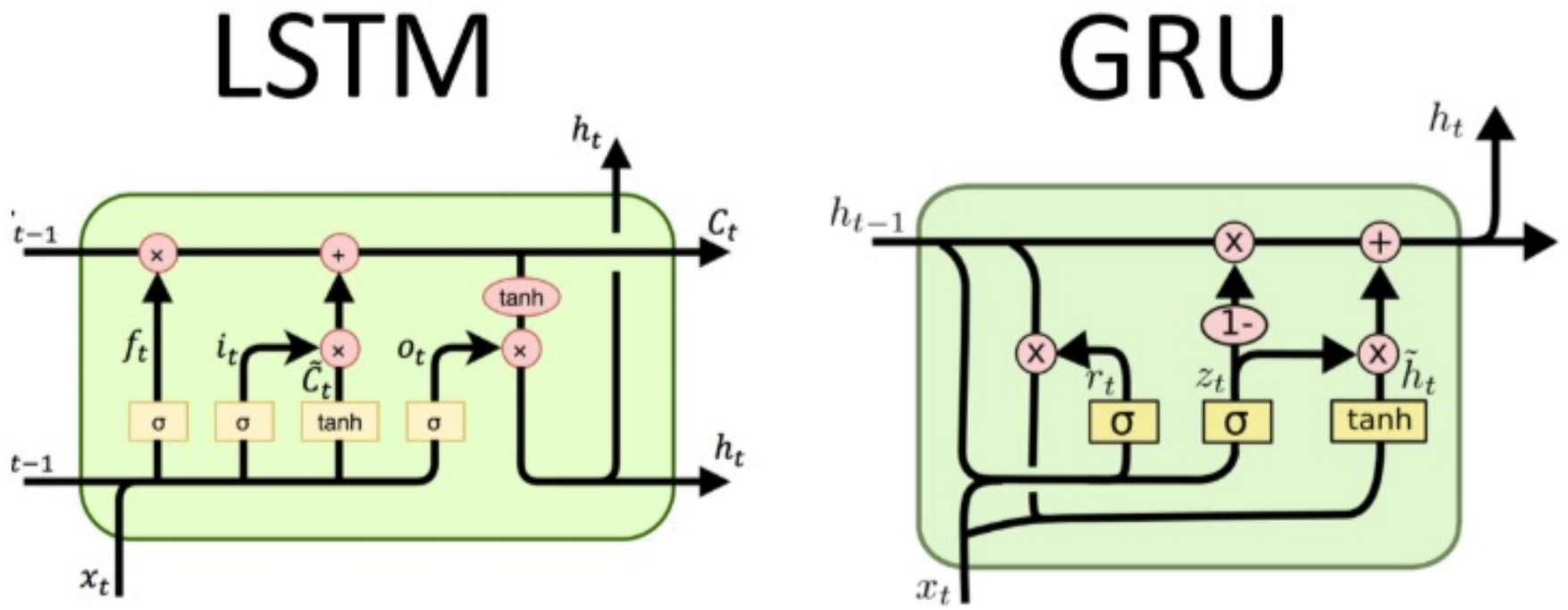
- Gated recurrent unit (GRU)
- Phased-LSTM

Gated recurrent units (GRUs)

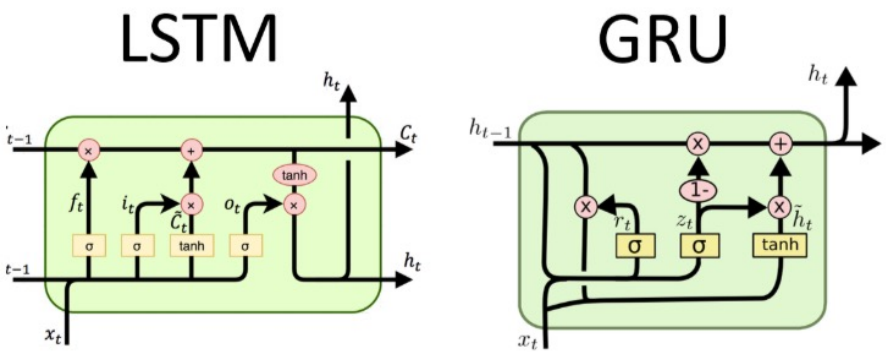
Gated recurrent units (GRUs) are newer (published in 2014) architecture and can be seen a simplification of LSTM (published in 1997).

- GRU is composed of only two gates: the reset gate and the update gate.
- It propagates a single state
- It's, then, the represented by fewer equations
  - → Less parameters to learn
  - → Faster to train

GRU vs LSTM



GRU vs LSTM



When do I prefer GRU over LSTM?

- It depends on the data
- It needs empirical evaluation

In this empirical study [2], GRU outperformed the LSTM on all tasks with the exception of language modelling.

[2] An Empirical Exploration of Recurrent Network Architectures. Jozefowic et al. (2015)

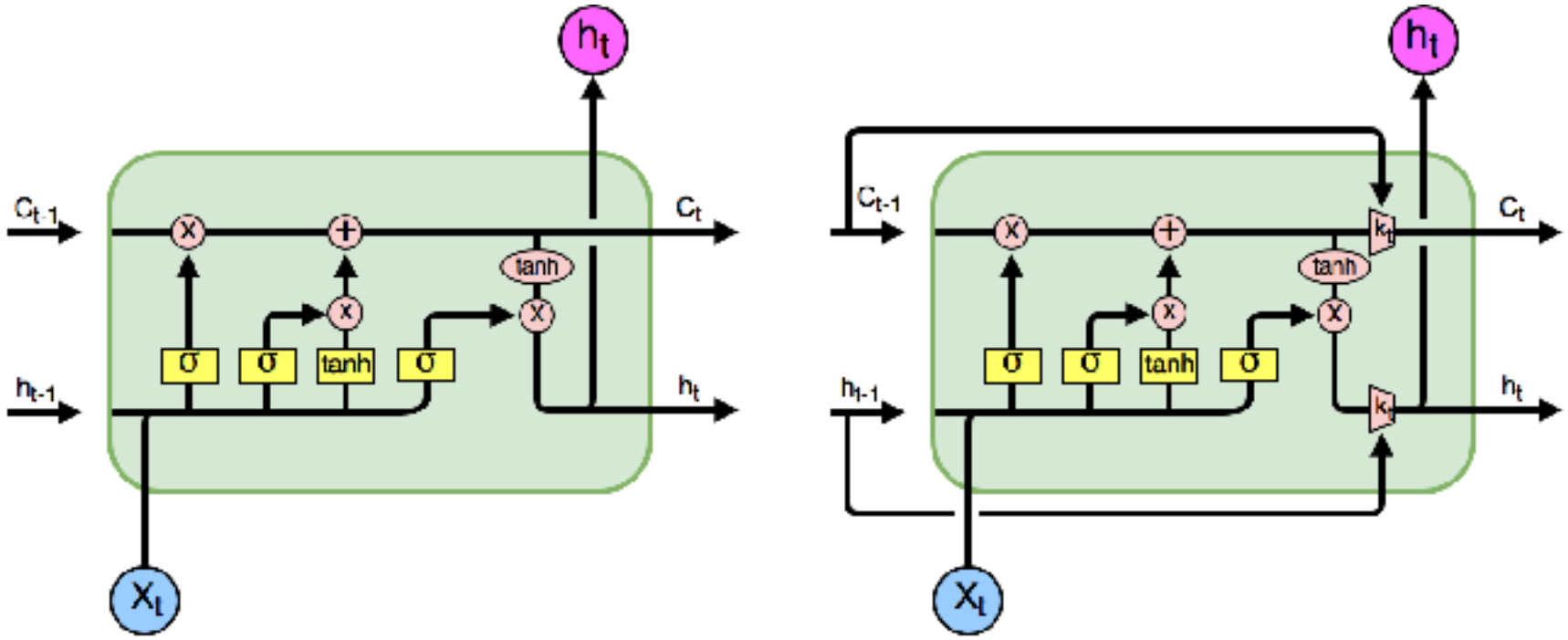
Phased-LSTM

Phased-LSTM [3] is designed to deal with input sampled at asynchronous times.

- Extends the LSTM unit by adding a new time gate.
- This gate is controlled by a parametrized oscillation with a frequency range that produces updates of the memory cell only during a small percentage of the cycle.
- The model naturally integrates inputs from sensors of arbitrary sampling rates.

[3] Neil, D., Pfeiffer, M., & Liu, S. C. (2016). Phased lstm: Accelerating recurrent network training for long or event-based sequences. *arXiv preprint arXiv:1610.09513*.

Phased-LSTM vs LSTM



(a) Standard LSTM

(b) Phased LSTM

Deep Learning for Time Series – Gated Models

Recap

In this lecture

- Long-term dependencies
    - Exploding/Vanishing gradient
- Long-short term memory networks (LSTMs)
- Other gated architectures
    - GRU
    - Phased-LSTM