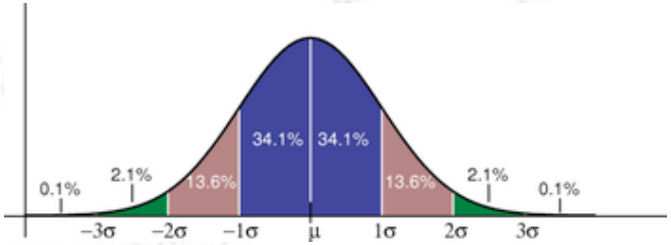
Applied ML

Long Short Term Memory (LSTM)

Troels C. Petersen (NBI)



"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

ML on time series

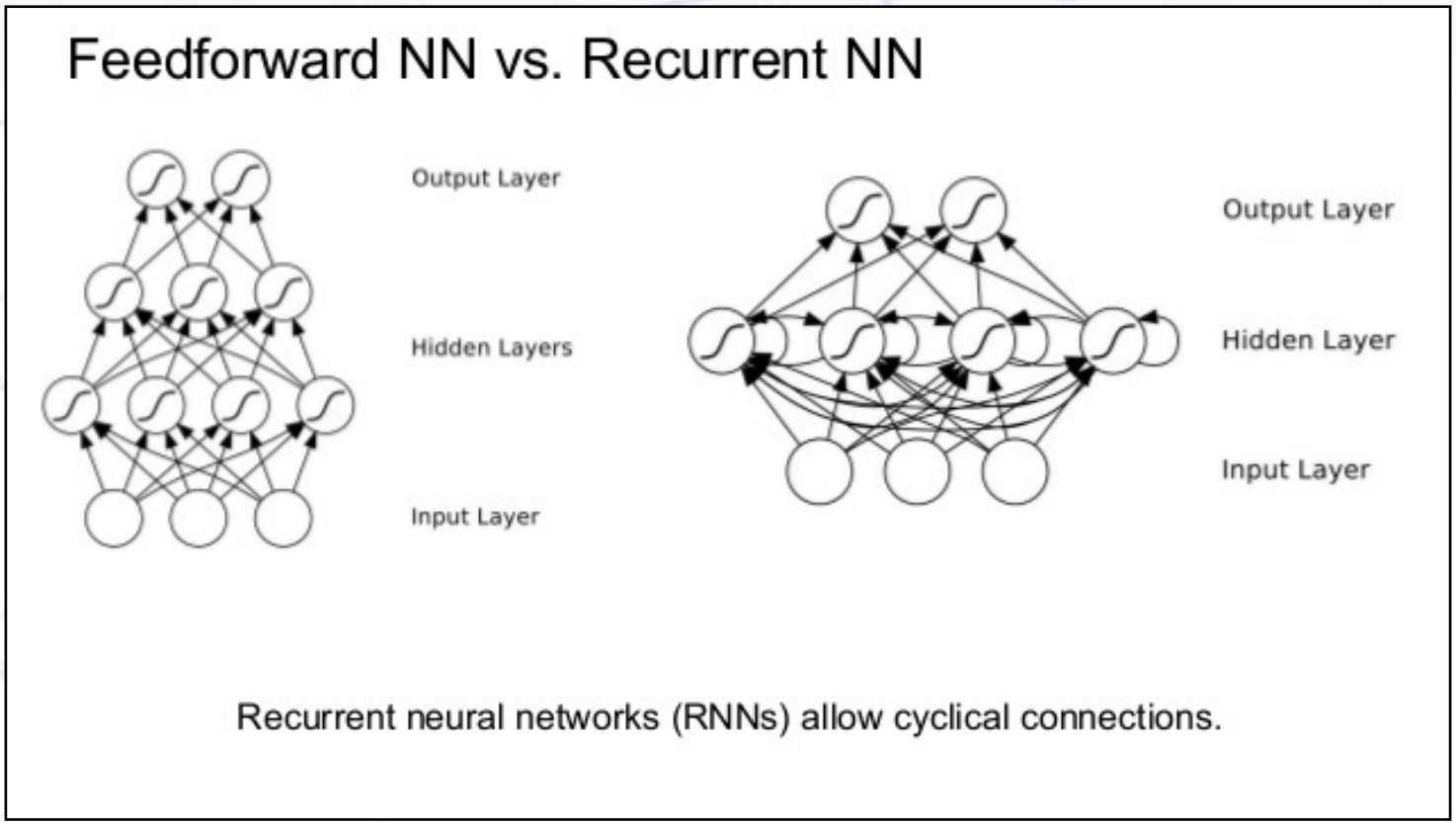
A lot of data manifest itself in the form of time series, where the task is to estimate the next value in the series. Examples: Text data, stock prices, etc.

Such data does not fit into the basic tree/forward NN paradigm, but requires special attention.

Recall: Recurrent NN

Normally, the information from one layer is fed forward to the next layer in a feedforward Neural Network (NN).

However, it may be of advantage to allow a network to give feedback, which is called a recurrent NN:

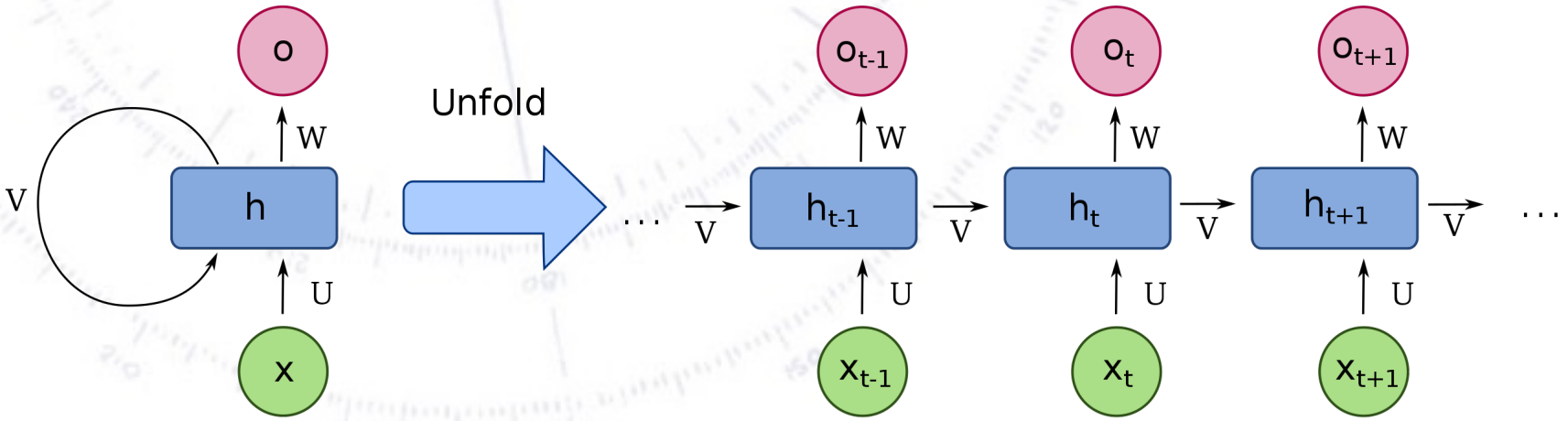


ML on time series

A lot of data manifest itself in the form of time series, where the task is to estimate the next value in the series. Examples: Text data, stock prices, etc.

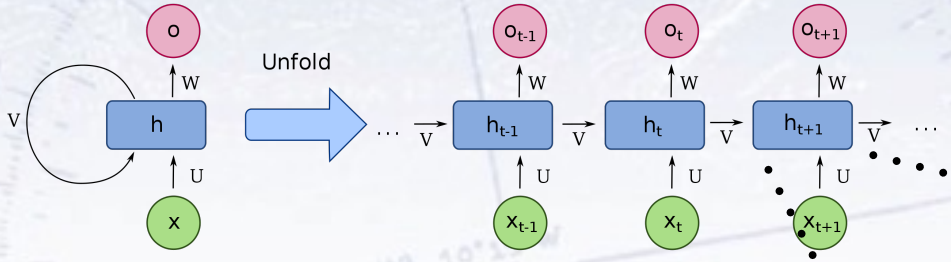
Such data does not fit into the basic tree/forward NN paradigm, but requires special attention.

One way to attack the problem is with a Recurrent Neural Network (RNN), where at each time step, the data is fed into the network along with the output of the hidden state:



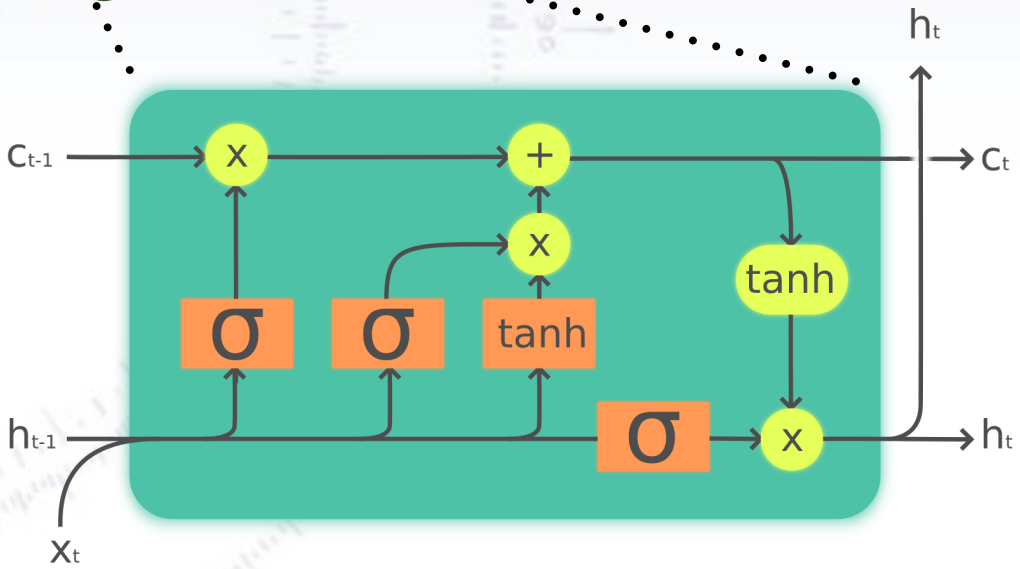
Long Short Term Memory

A special kind of RNN is the Long Short Term Memory (LSTM) network.

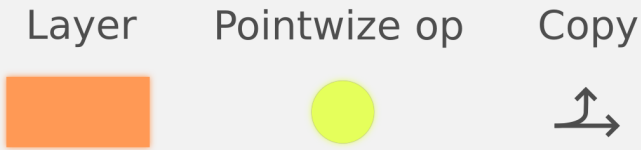


Each unit gets as input the new input features $\mathbf{x_t}$, along with the cell state $\mathbf{c_{t-1}}$ and the hidden state (which also contains predictions!) $\mathbf{h_{t-1}}$.

These are combined as shown, and a new cell states $\mathbf{c_t}$ and hidden state $\mathbf{h_t}$ is produced.

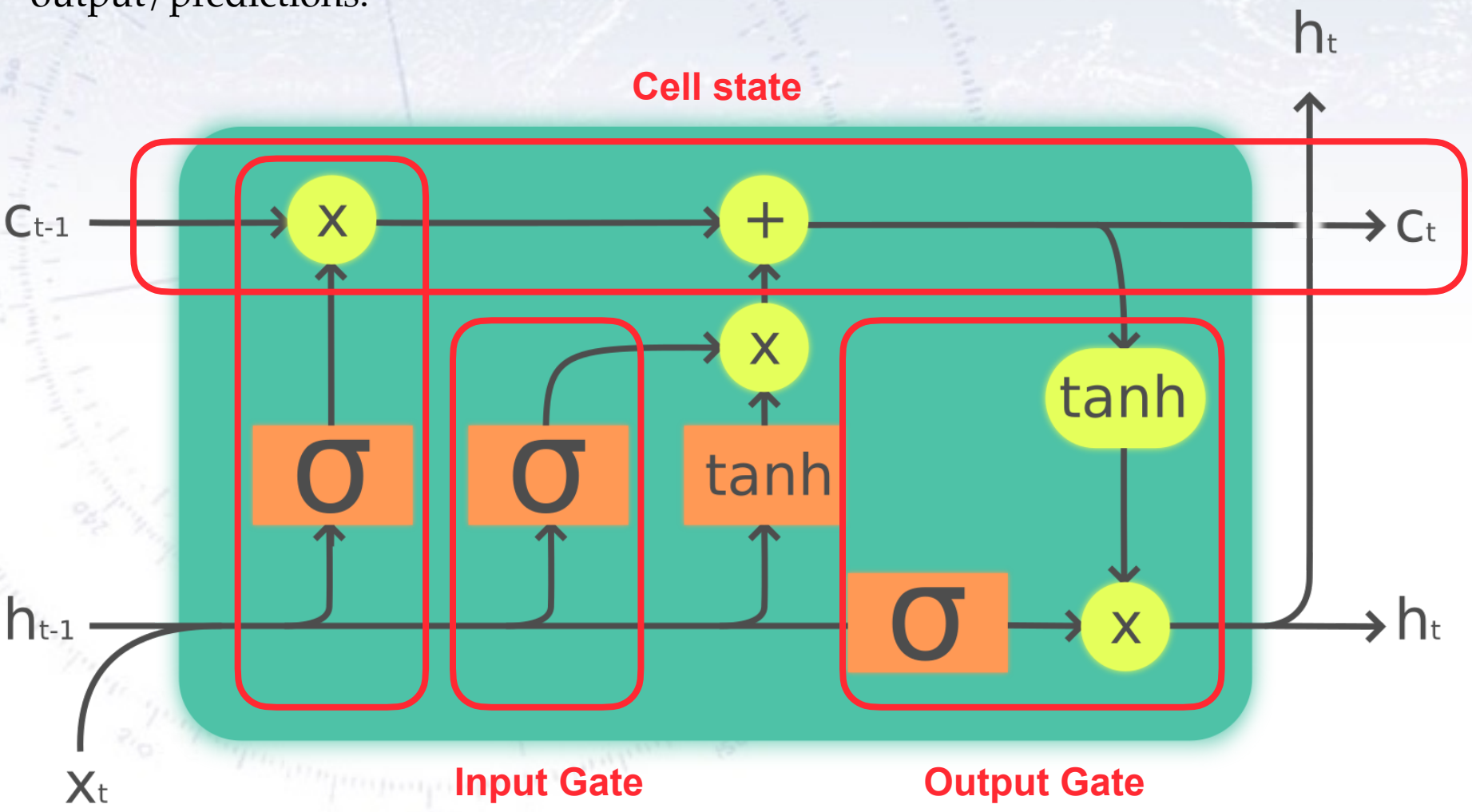


Legend:



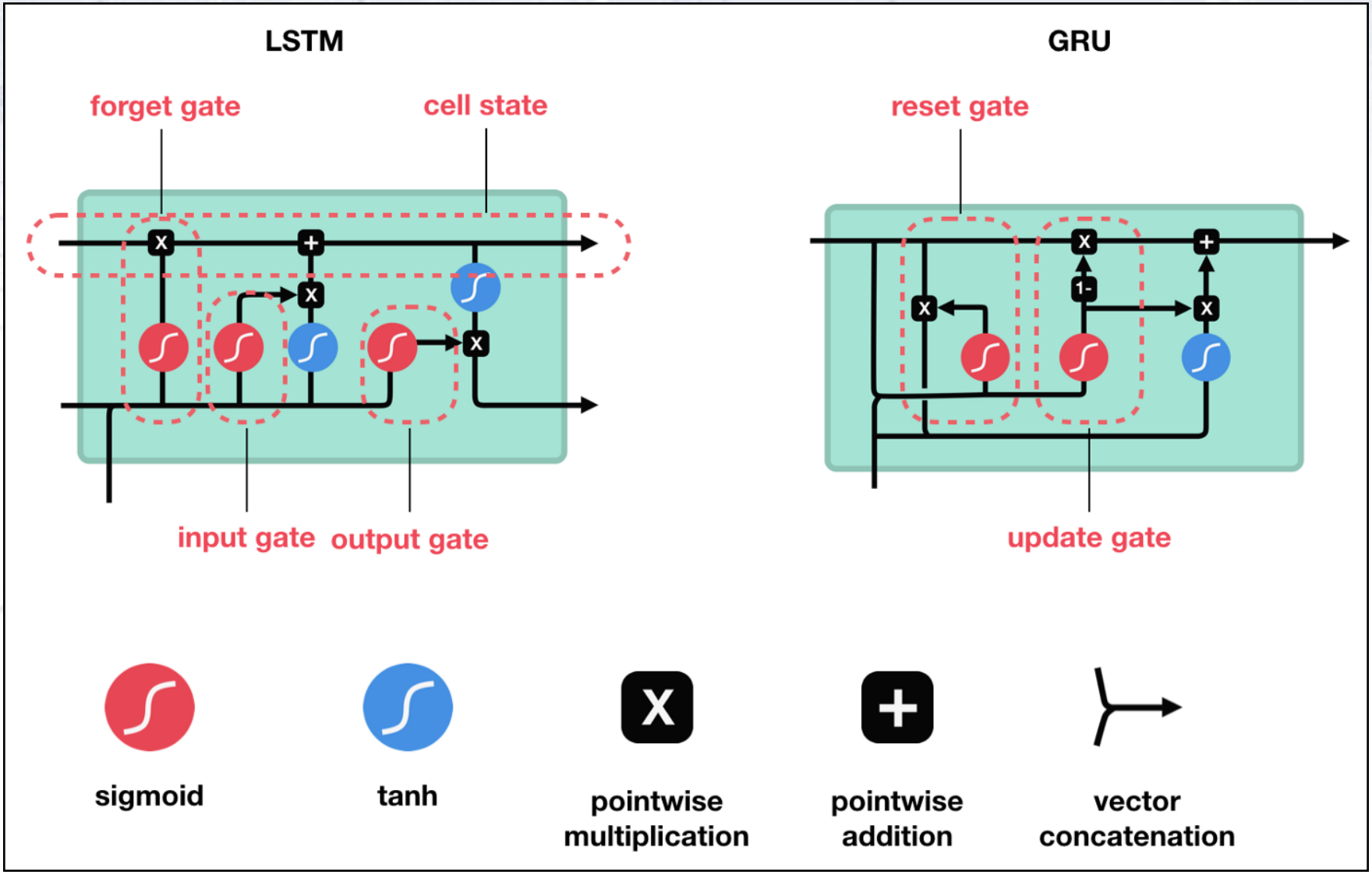
Long Short Term Memory

The LSTM consists of different gates, which operate together to give the best output/predictions.



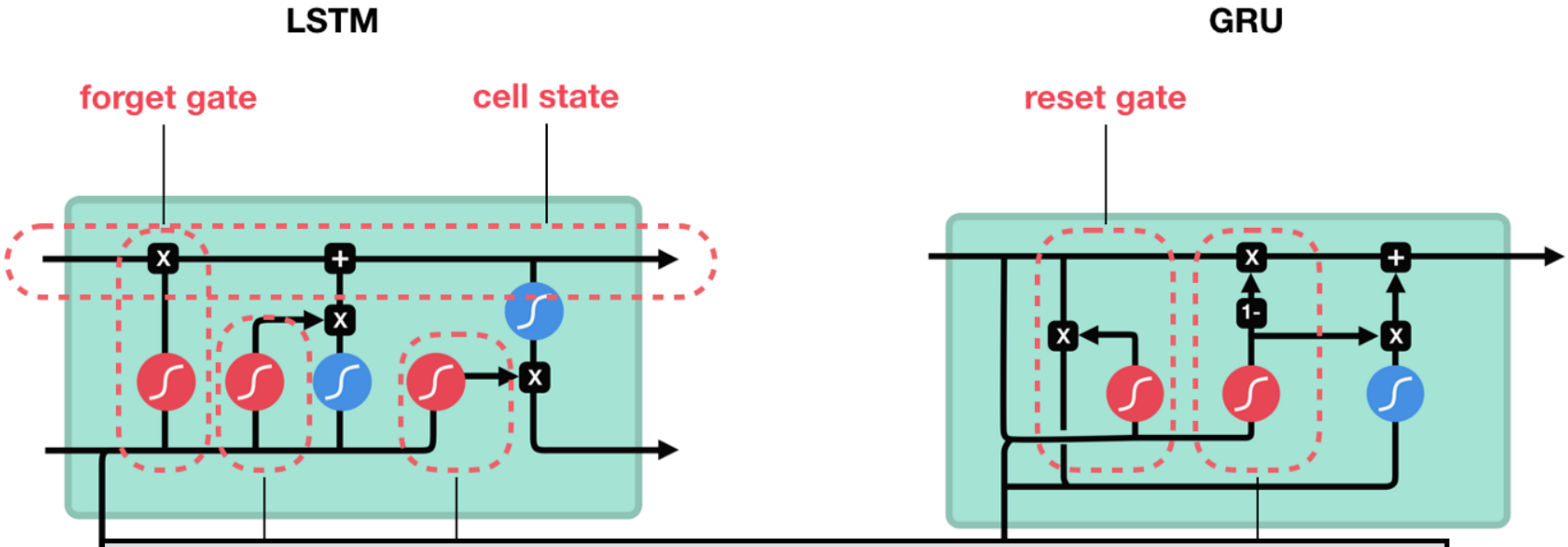
Gated Recurrent Unit (GRU)

A GRU is a newer and simpler RNN, which uses hidden state for info transfer.



Gated Recurrent Unit (GRU)

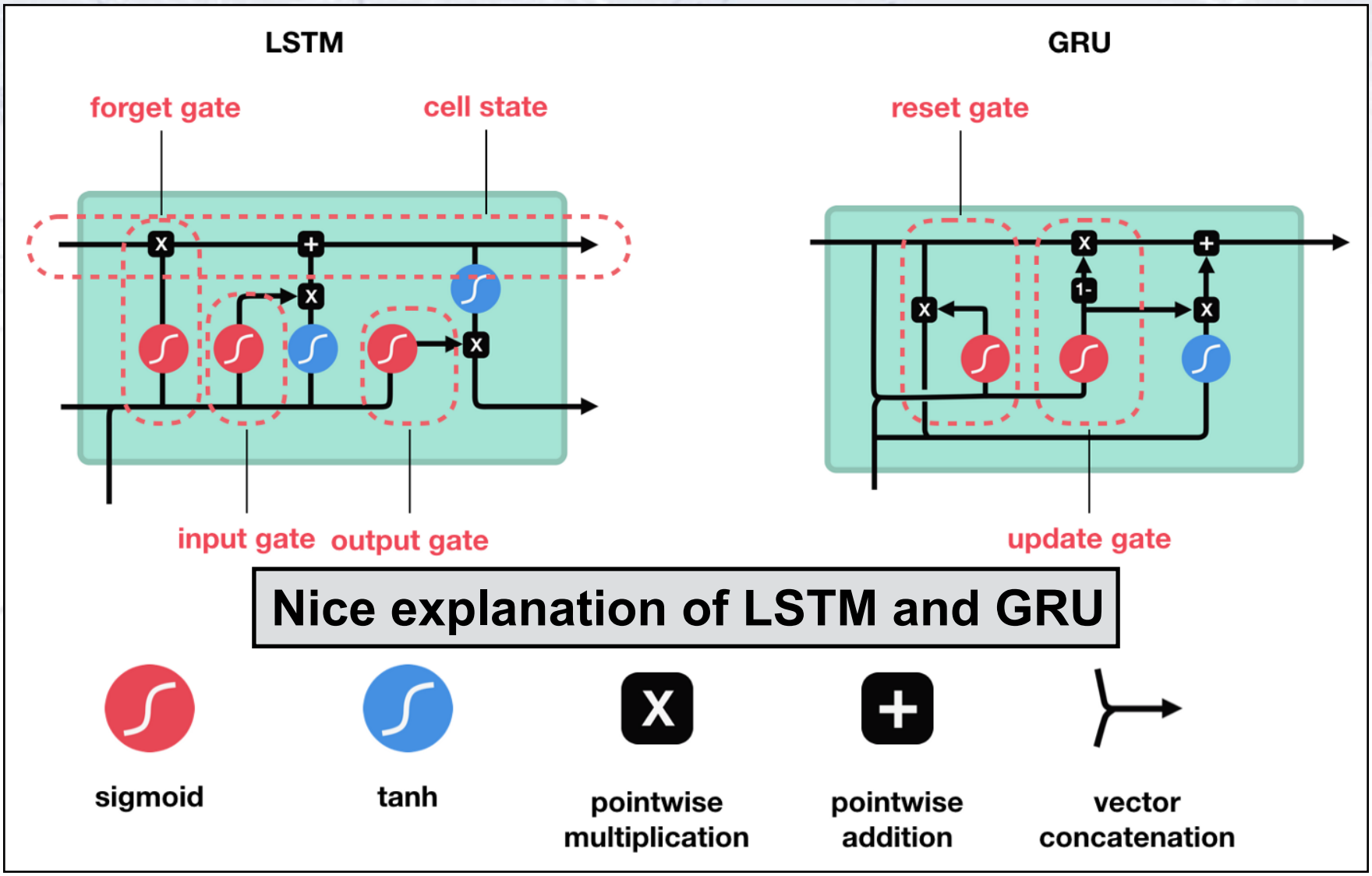
A GRU is a newer and simpler RNN, which uses hidden state for info transfer.



"Almost all state of the art results based on recurrent neural networks are achieved with these two networks"
[Towards data science]

Gated Recurrent Unit (GRU)

A GRU is a newer and simpler RNN, which uses hidden state for info transfer.



Back propagation

Recall, that **back propagation** is the modification of NN weights to **minimise the error** of the network output compared to the target values.

The algorithm of **gradient descent** works as follows:
1. Present a training input pattern and use it to get an output.
2. Compare the predicted to the expected outputs and calculate the error.
3. Calculate the derivatives of the error with respect to the network weights.
4. Adjust the weights to minimise the error.

Repeating this should make the **weights converge towards optimal values**.

Note that the derivatives typically are calculated **for a batch of training cases**, to avoid updating the weights before the statistical fluctuations are small, but still large enough to provide some degree of stochastic fluctuations.

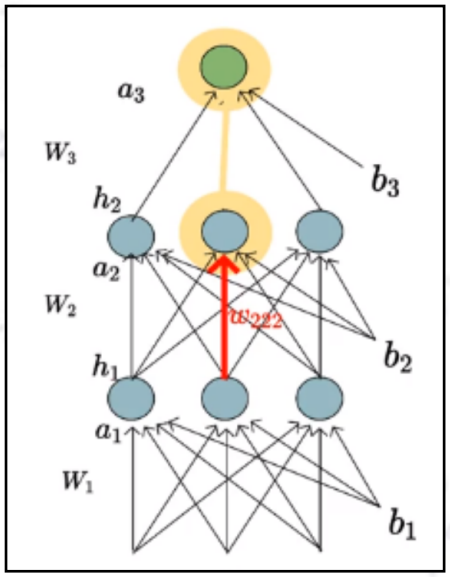
Back propagation

Recall, that **back propagation** is the modification of NN weights to **minimise the error** of the network output compared to the target values.

The algorithm of **gradient descent** works as follows:

1. Present a training input pattern and use it to get an output.
2. Compare the predicted to the expected outputs and calculate the error.
3. Calculate the derivatives of the error with respect to the network weights.
4. Adjust the weights to minimise the error.

Repeating this should make the **weights converge towards optimal values**.



$$(w_{222})_{t+1} = (w_{222})_t - \eta * \left(\frac{\partial L}{\partial w_{222}}\right)$$

$$\frac{\partial L}{\partial w_{222}} = \left(\frac{\partial L}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right)$$

$$= \left(\frac{\partial L}{\partial h_{22}}\right).\left(\frac{\partial h_{22}}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right)$$

$$= \left(\frac{\partial L}{\partial a_{31}}\right).\left(\frac{\partial a_{31}}{\partial h_{22}}\right).\left(\frac{\partial h_{22}}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right)$$

$$= \left(\frac{\partial L}{\partial \hat{y}}\right).\left(\frac{\partial \hat{y}}{\partial a_{31}}\right).\left(\frac{\partial a_{31}}{\partial h_{22}}\right).\left(\frac{\partial h_{22}}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right)$$

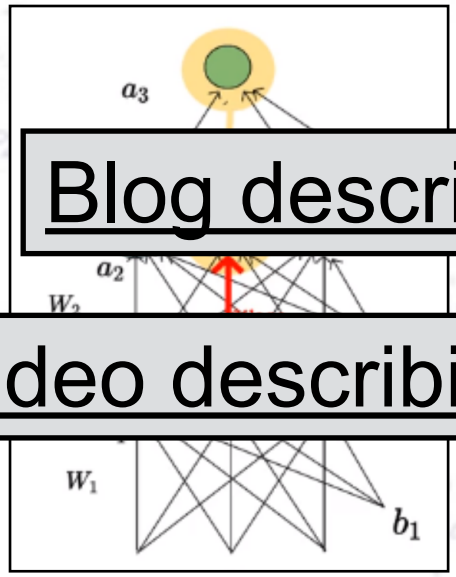
Back propagation

Recall, that **back propagation** is the modification of NN weights to **minimise the error** of the network output compared to the target values.

The algorithm of **gradient descent** works as follows:

1. Present a training input pattern and use it to get an output.
2. Compare the predicted to the expected outputs and calculate the error.
3. Calculate the derivatives of the error with respect to the network weights.
4. Adjust the weights to minimise the error.

Repeating this should make the **weights converge towards optimal values**.



$$(w_{222})_{t+1} = (w_{222})_t - \eta * (\frac{\partial L}{\partial w_{222}})$$

$$= (\frac{\partial L}{\partial h_{22}}).(\frac{\partial h_{22}}{\partial a_{22}}).(\frac{\partial a_{22}}{\partial w_{222}})$$

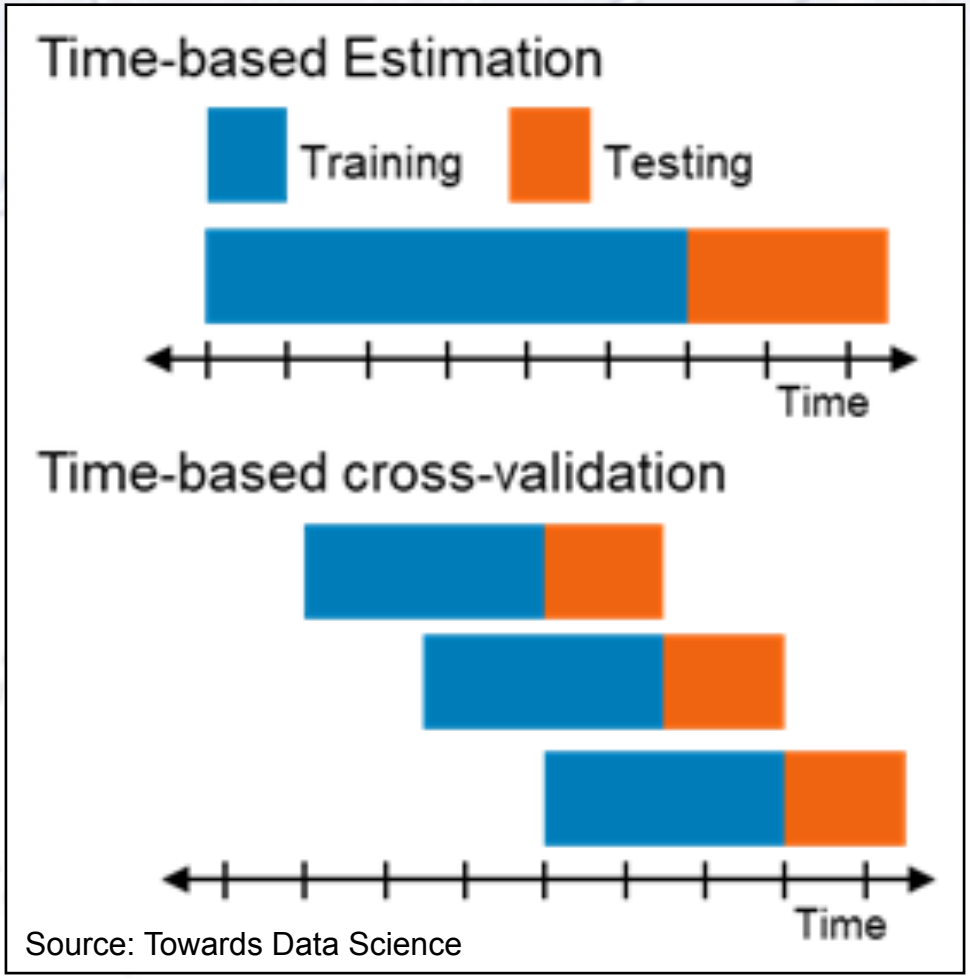
$$= (\frac{\partial L}{\partial \hat{y}}).(\frac{\partial \hat{y}}{\partial a_{31}}).(\frac{\partial a_{31}}{\partial h_{22}}).(\frac{\partial h_{22}}{\partial a_{22}}).(\frac{\partial a_{22}}{\partial w_{222}})$$

Blog describing back propagation nicely

Video describing back propagation very nicely

Training & Testing for LSTMs

In order for an LSTM to work "in real life", one has to ensure that one does not use input data "from the future". This is done by splitting in time (not random!).



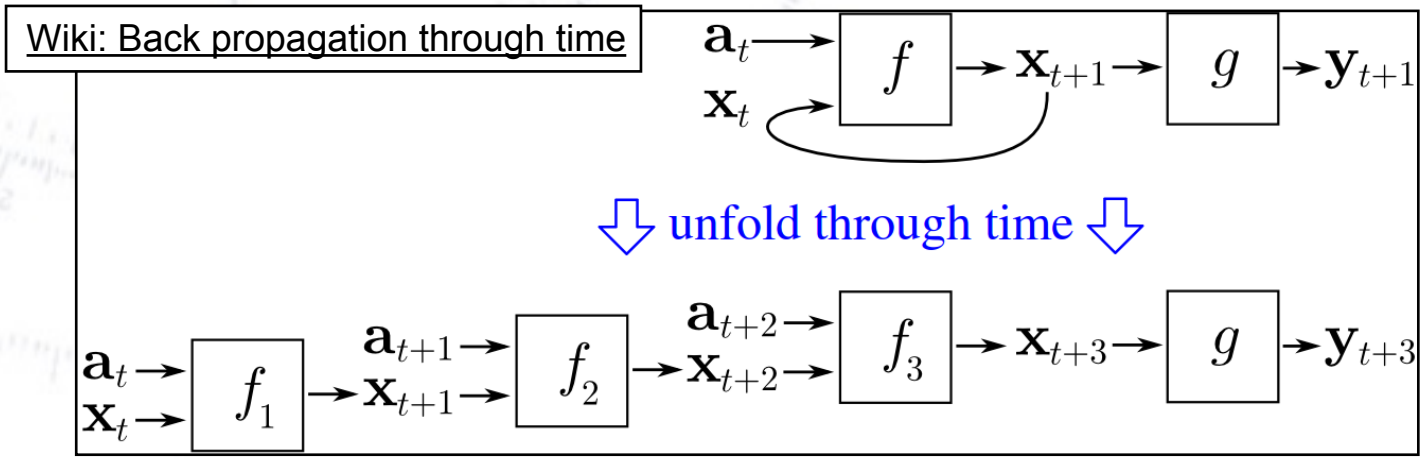
Source: Towards Data Science

Back propagation through time

In **back propagation through time** (for e.g. LSTM), the weights are modified by "unrolling" all input timesteps. Each timestep has one input timestep, one copy of the network, and one output. Errors are then calculated and accumulated for each timestep. The network is rolled back up and the weights are updated.

Spatially, each timestep of the unrolled recurrent neural network may be seen as an additional layer. In summary, the BPTT algorithm does as follows:

1. Present a sequence of timesteps of input and output pairs to the network.
2. Unroll the network then calculate and accumulate errors across each timestep.
3. Roll-up the network and update weights.
4. Repeat.



Status of LSTMs

In 2018, OpenAI also trained a similar LSTM by policy gradients to control a human-like robot hand that manipulates physical objects with unprecedented dexterity [skill in performing task with hands].

In 2019, DeepMind's program AlphaStar used a deep LSTM core to excel at the complex video game Starcraft II. This was viewed as significant progress towards Artificial General Intelligence.

Status of LSTMs