Recurrent Neural Networks

Bruno Gonçalves
www.bgoncalves.com
github.com/bmtgoncalves/RNN

NYU DATA SCIENCE

Recurr Networks

Bruno G

www.bgo

github.com NN

JPMorgan Chase & Co.

Recurrent Neural Networks

Bruno Gonçalves

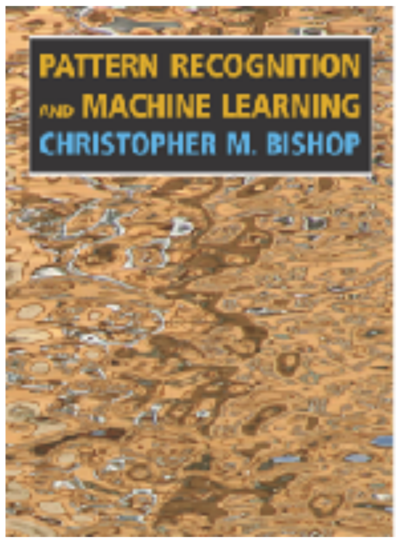
www.bgoncalves.com

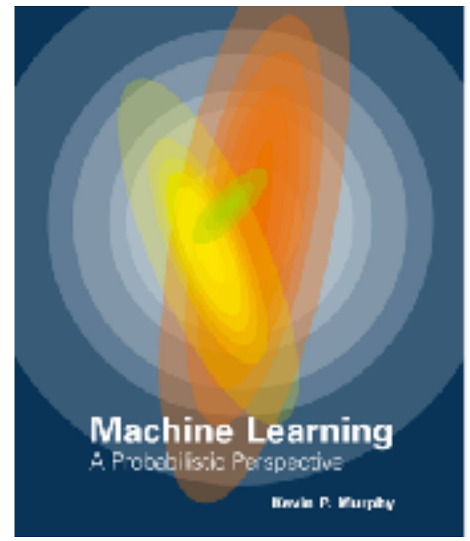
github.com/bmtgoncalves/RNN

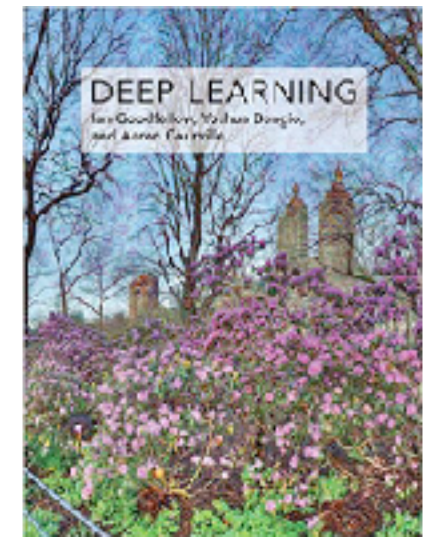
Disclaimer

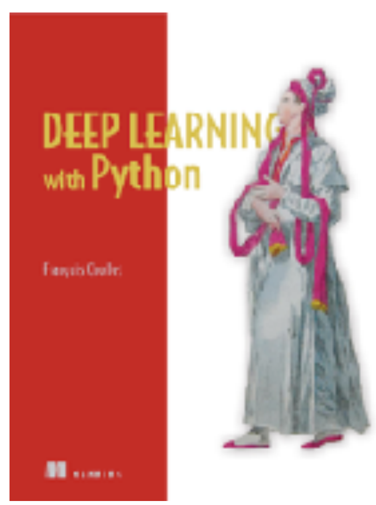
The views and opinions expressed in this article are those of the authors and do not necessarily reflect the official policy or position of my employer. The examples provided with this tutorial were chosen for their didactic value and are not mean to be representative of my day to day work.

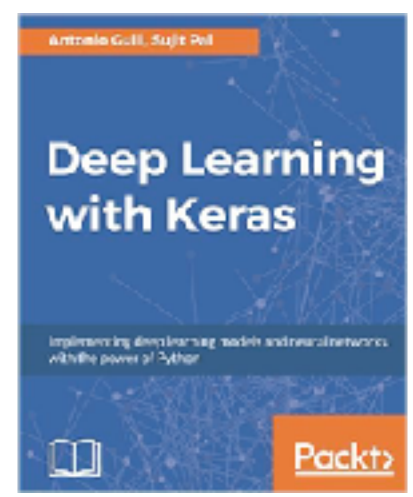
References


PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

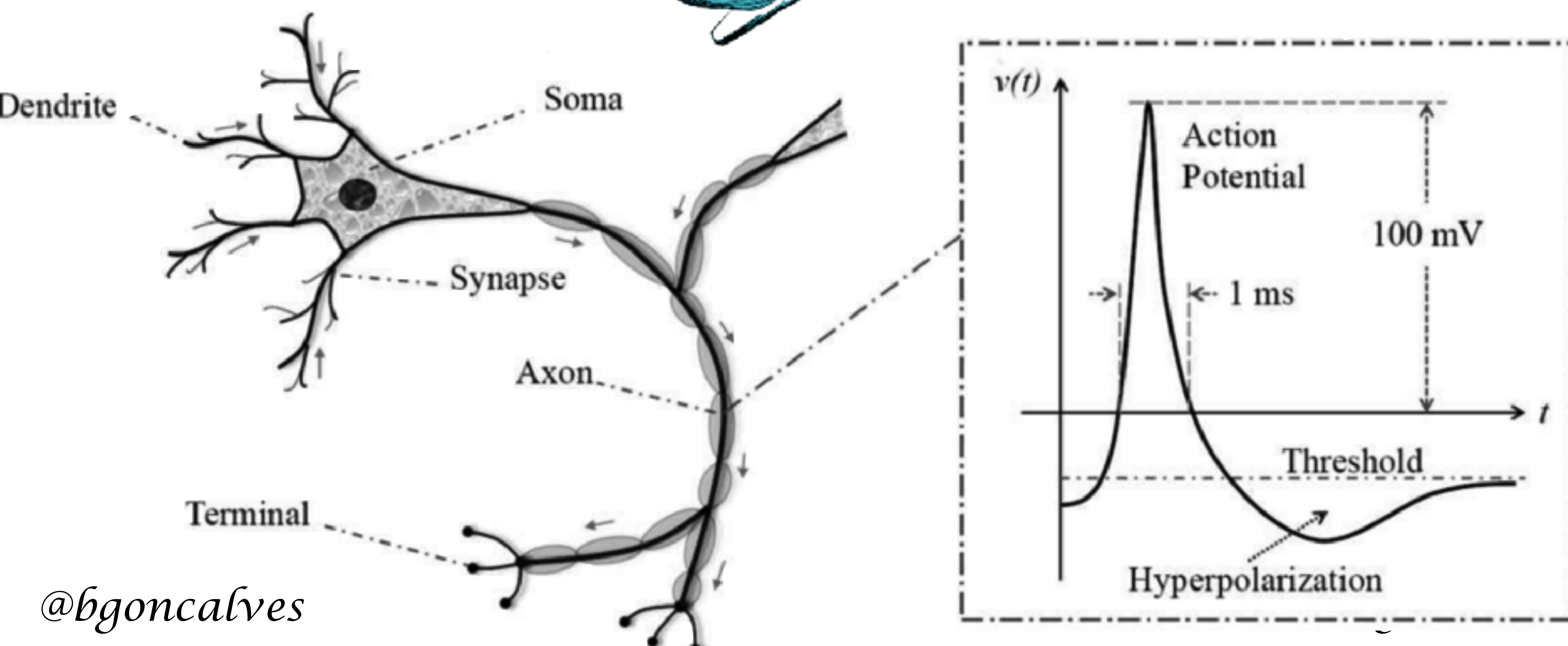

Machine Learning
A Probabilistic Perspective
Kevin P. Murphy


DEEP LEARNING


DEEP LEARNING
with Python

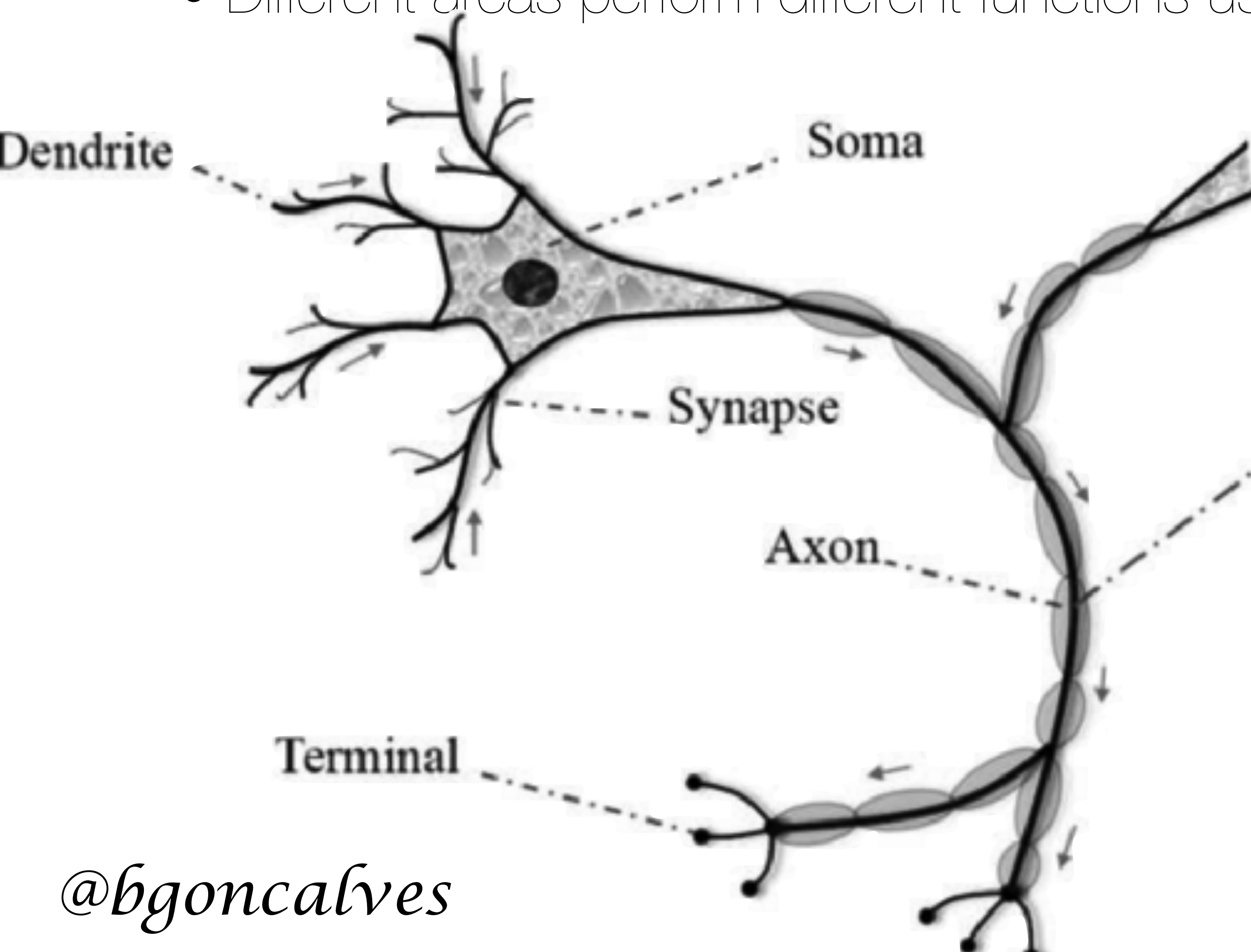

Antonio Gulli, Sujit Pal
Deep Learning
with Keras
Packt>

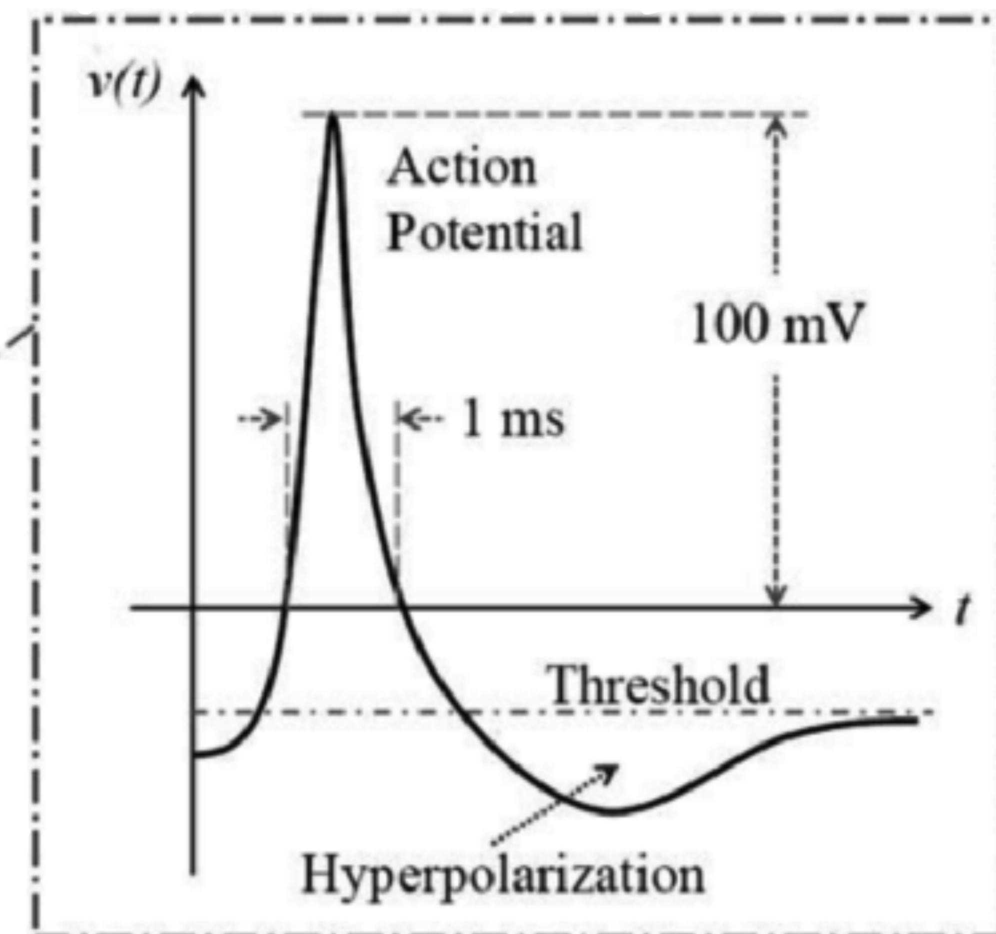
How the Brain "Works" (Cartoon version)



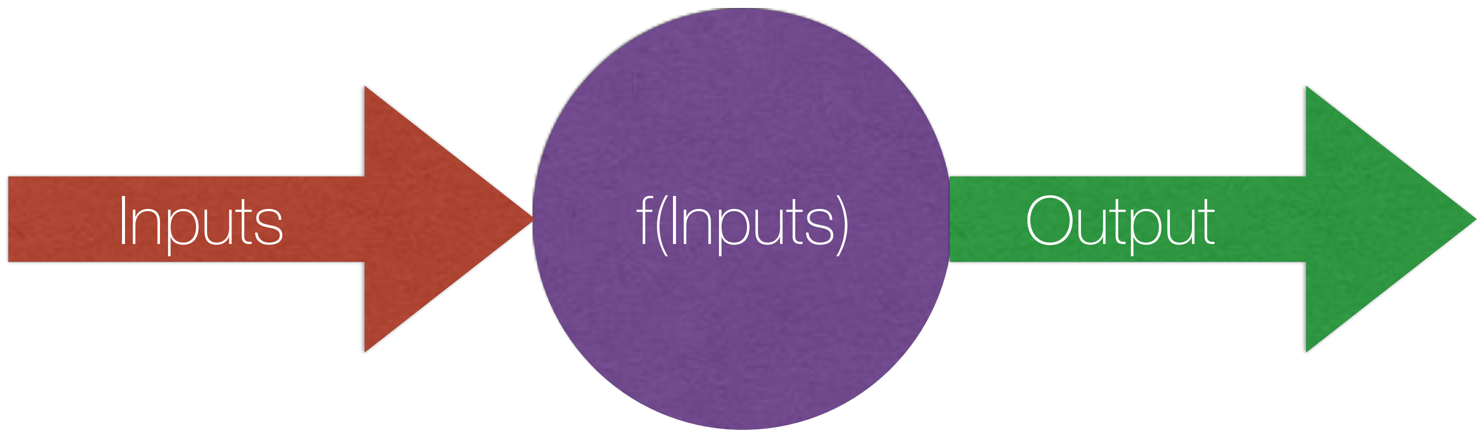
How the Brain “Works” (Cartoon version)

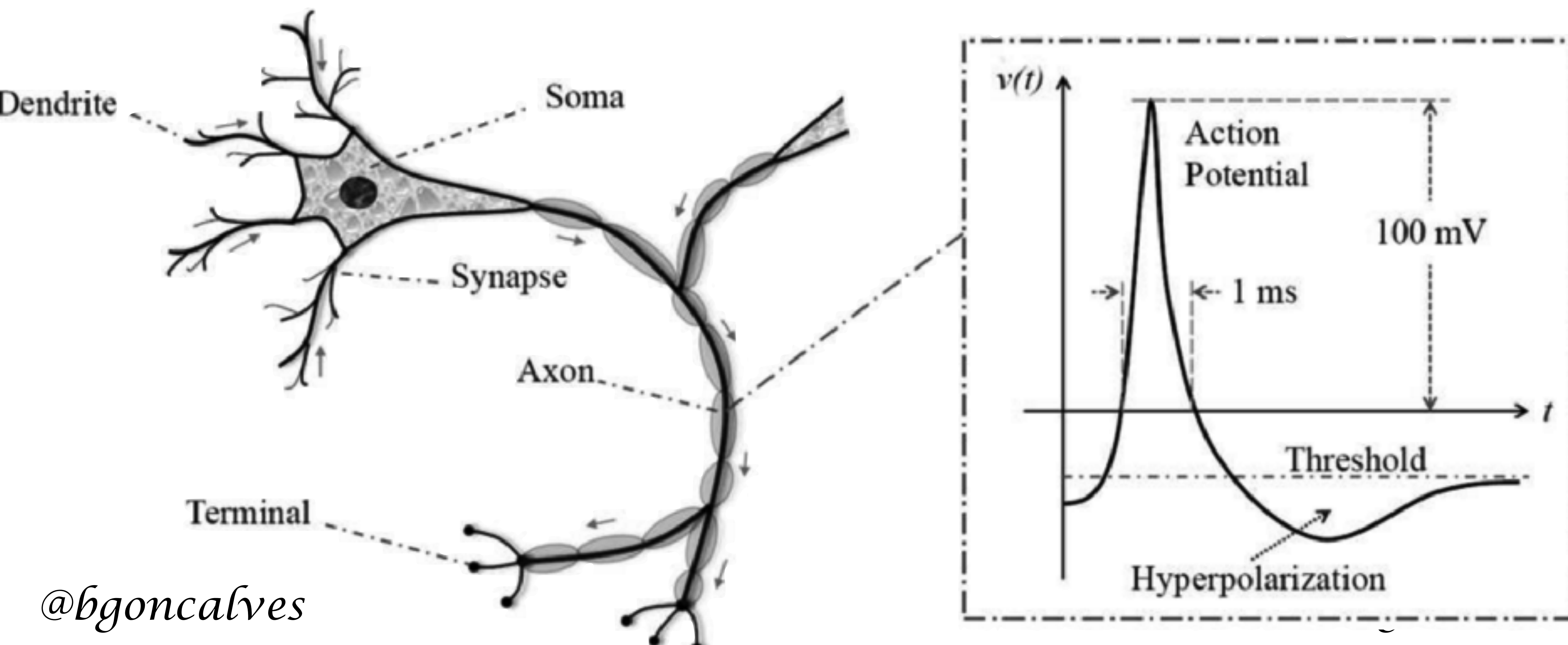
- Each neuron receives input from other neurons
- 10^{11} neurons, each with with 10^4 weights
- Weights can be positive or negative
- Weights adapt during the learning process
- “neurons that fire together wire together” (Hebb)
- Different areas perform different functions using same structure (Modularity)





How the Brain "Works" (Cartoon version)

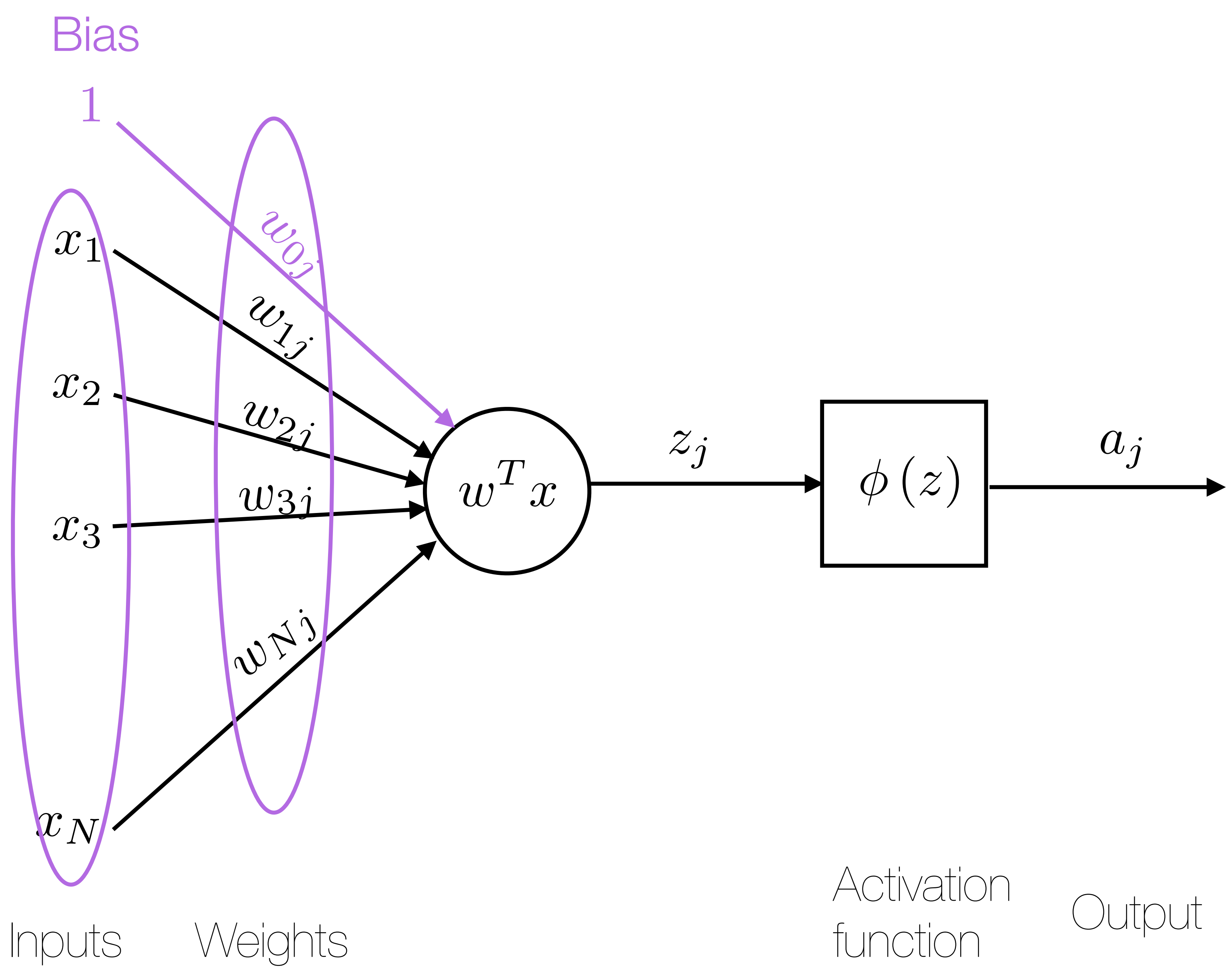




Optimization Problem

- (Machine) Learning can be thought of as an optimization problem.
- Optimization Problems have 3 distinct pieces:
  - The constraints — Neural Network
  - The function to optimize — Prediction Error
  - The optimization algorithm. — Gradient Descent

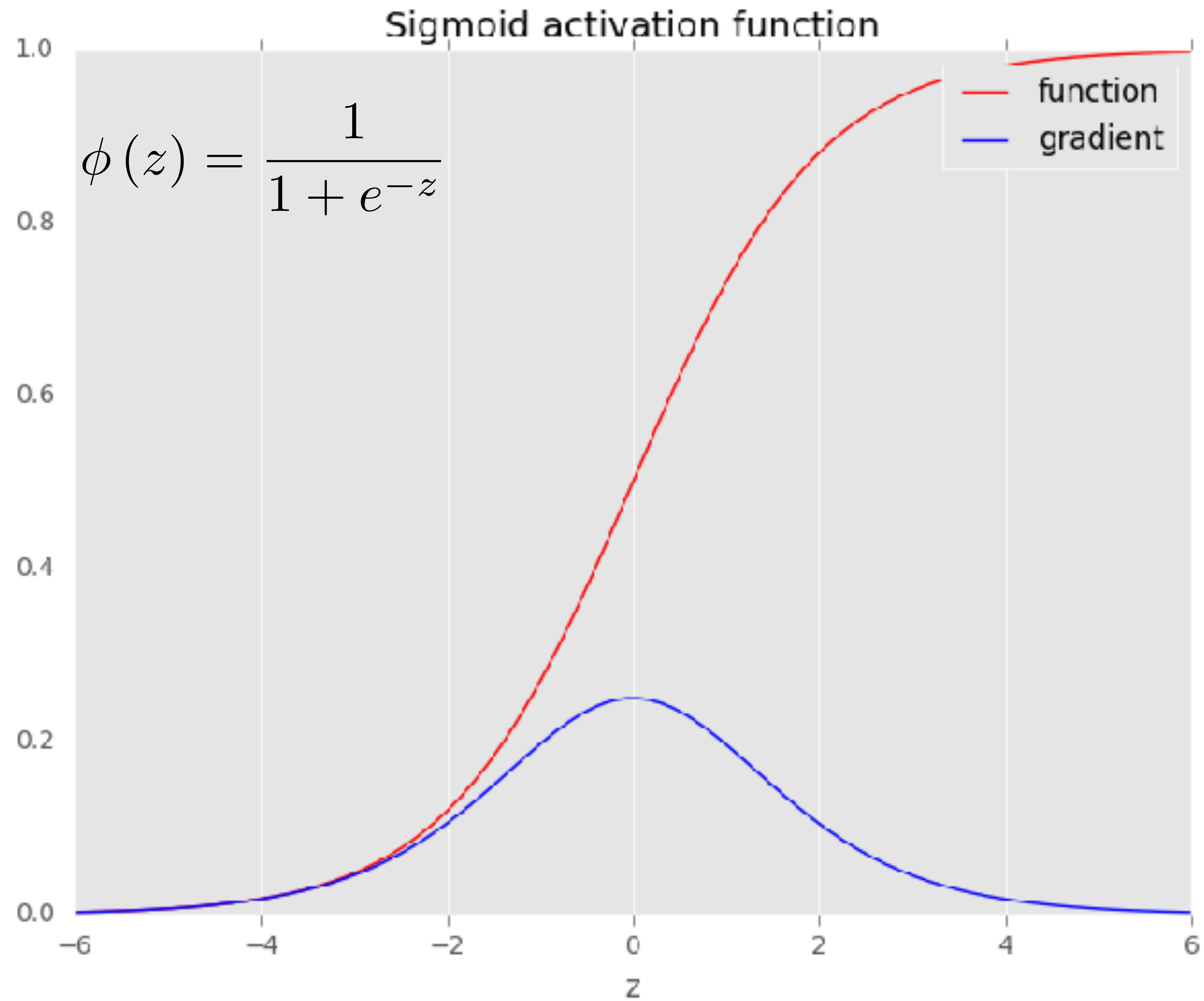
Artificial Neuron



Activation Function - Sigmoid

http://github.com/bmtgoncalves/Neural-Networks

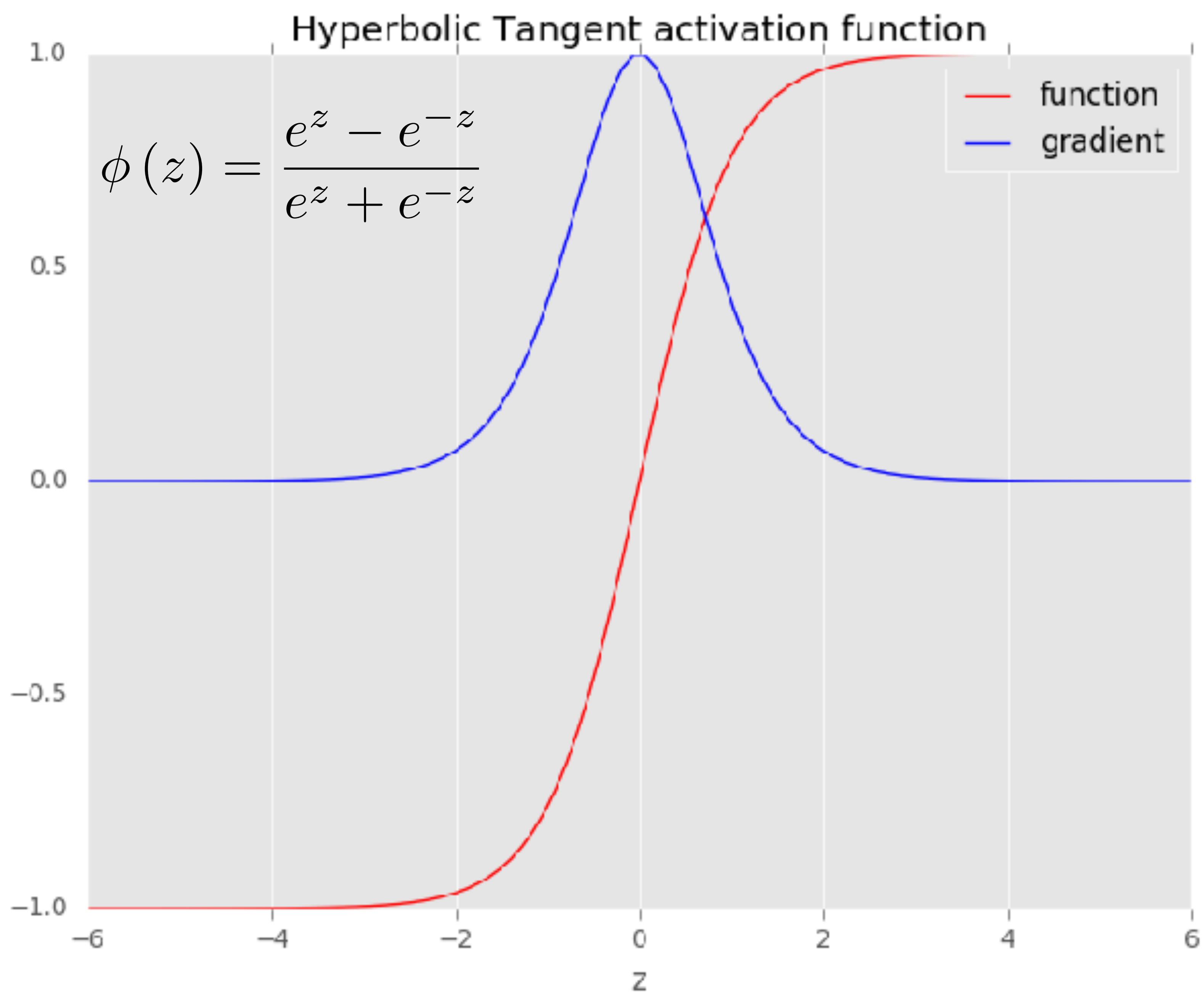
- Non-Linear function
- Differentiable
- non-decreasing
- Compute new sets of features
- Each layer builds up a more abstract representation of the data
- Perhaps the most common



Activation Function - tanh

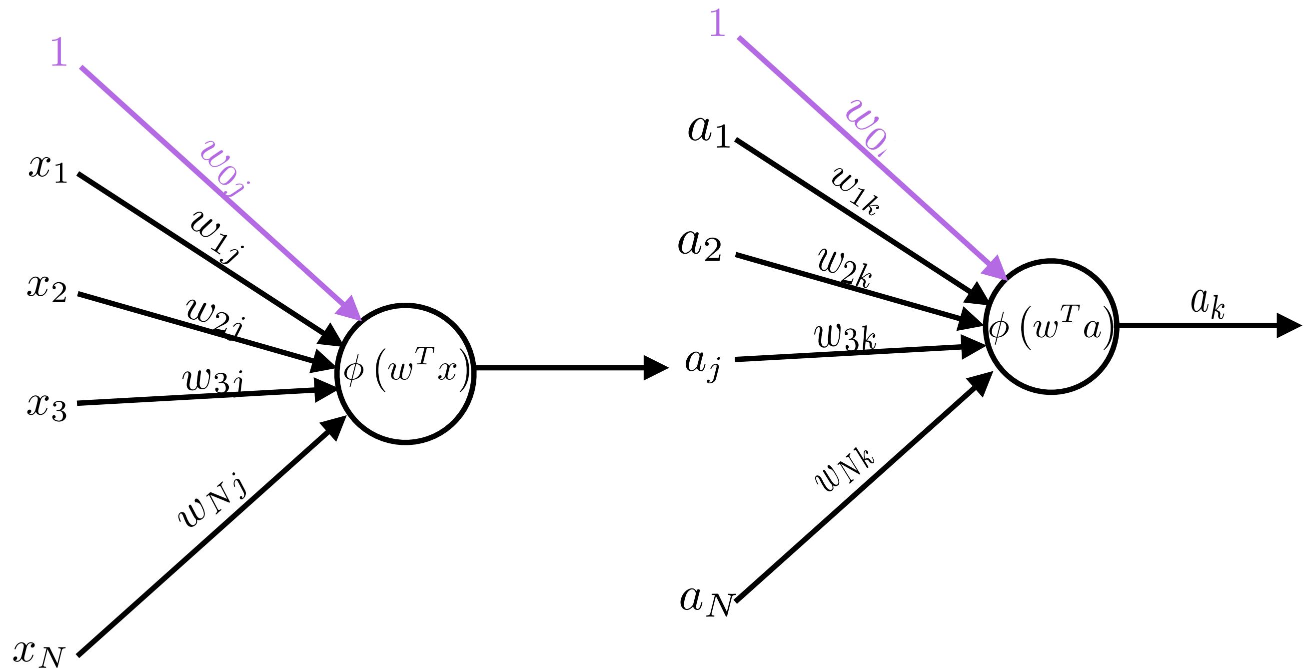
http://github.com/bmtgoncalves/Neural-Networks

- Non-Linear function
- Differentiable
- non-decreasing
- Compute new sets of features
- Each layer builds up a more abstract representation of the data



Forward Propagation

- The output of a perceptron is determined by a sequence of steps:
    - obtain the inputs
    - multiply the inputs by the respective weights
    - calculate output using the activation function
- To create a multi-layer perceptron, you can simply use the output of one layer as the input to the next one.



- But how can we propagate back the errors and update the weights?

Backward Propagation of Errors (BackProp)

- BackProp operates in two phases:
  - Forward propagate the inputs and calculate the deltas
  - Update the weights
- The error at the output is a weighted average difference between predicted output and the observed one.
- For inner layers there is no “real output”!

Loss Functions

- For learning to occur, we must quantify how far off we are from the desired output. There are two common ways of doing this:

  - Quadratic error function:

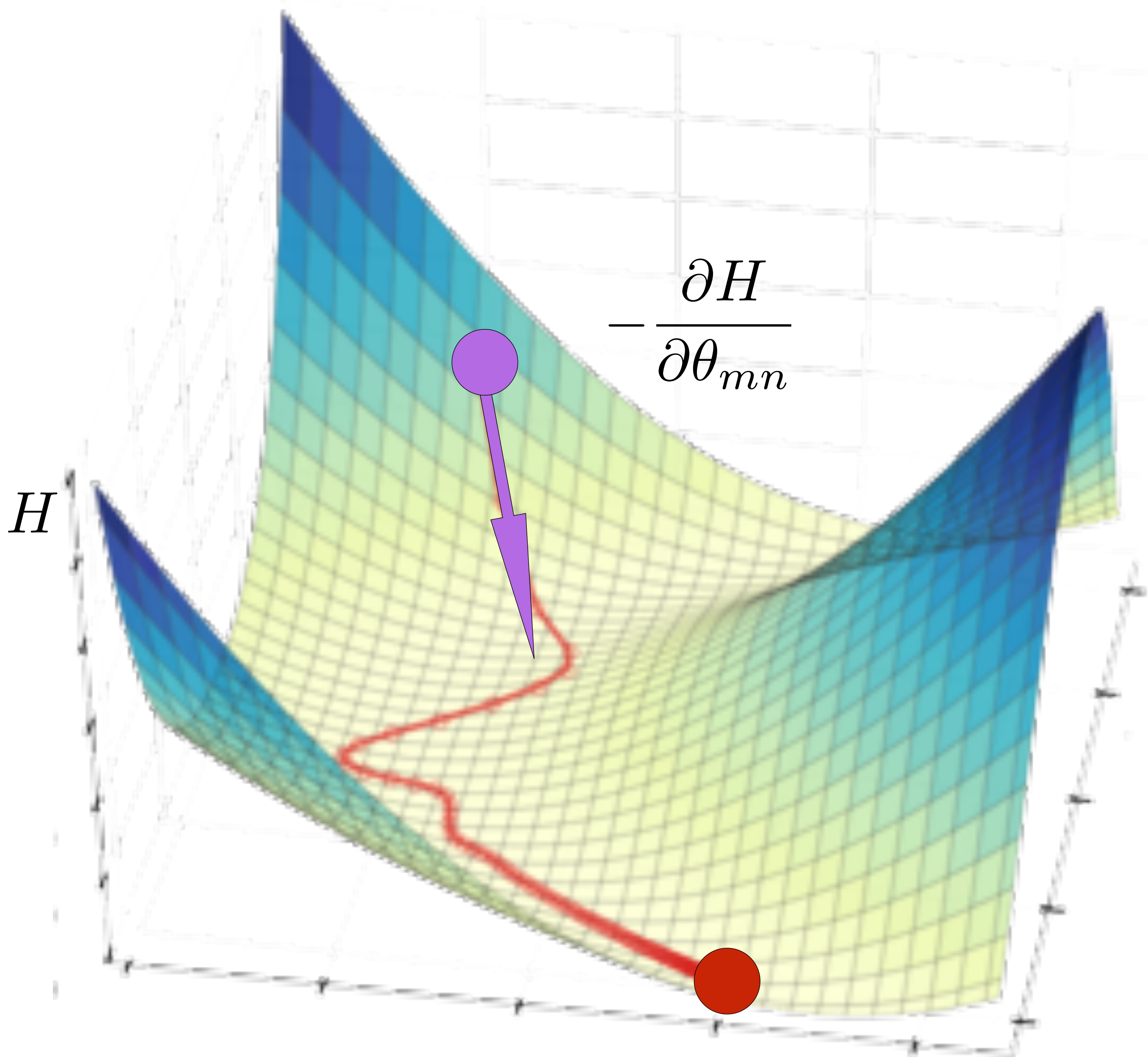
$$E = \frac{1}{N} \sum_n |y_n - a_n|^2$$

  - Cross Entropy

$$J = -\frac{1}{N} \sum_n \left[y_n^T \log a_n + (1 - y_n)^T \log (1 - a_n) \right]$$

The Cross Entropy is complementary to sigmoid activation in the output layer and improves its stability.

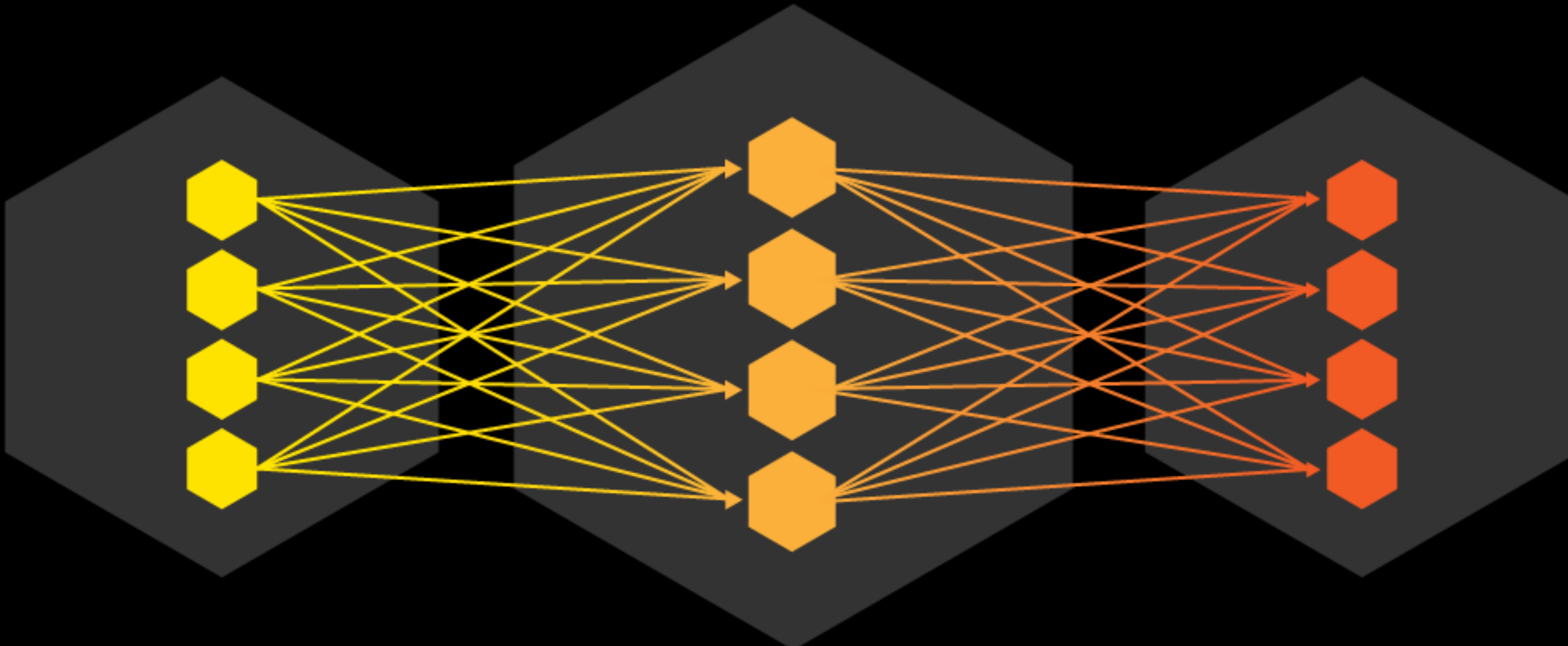
Gradient Descent



- Find the gradient for each training batch
- Take a step **downhill** along the direction of the gradient

$$\theta_{mn} \leftarrow \theta_{mn} - \alpha \frac{\partial H}{\partial \theta_{mn}}$$

- where α is the step size.
- Repeat until "convergence".



INPUT TERMS

FEATURES
PREDICTIONS
ATTRIBUTES
PREDICTABLE VARIABLES

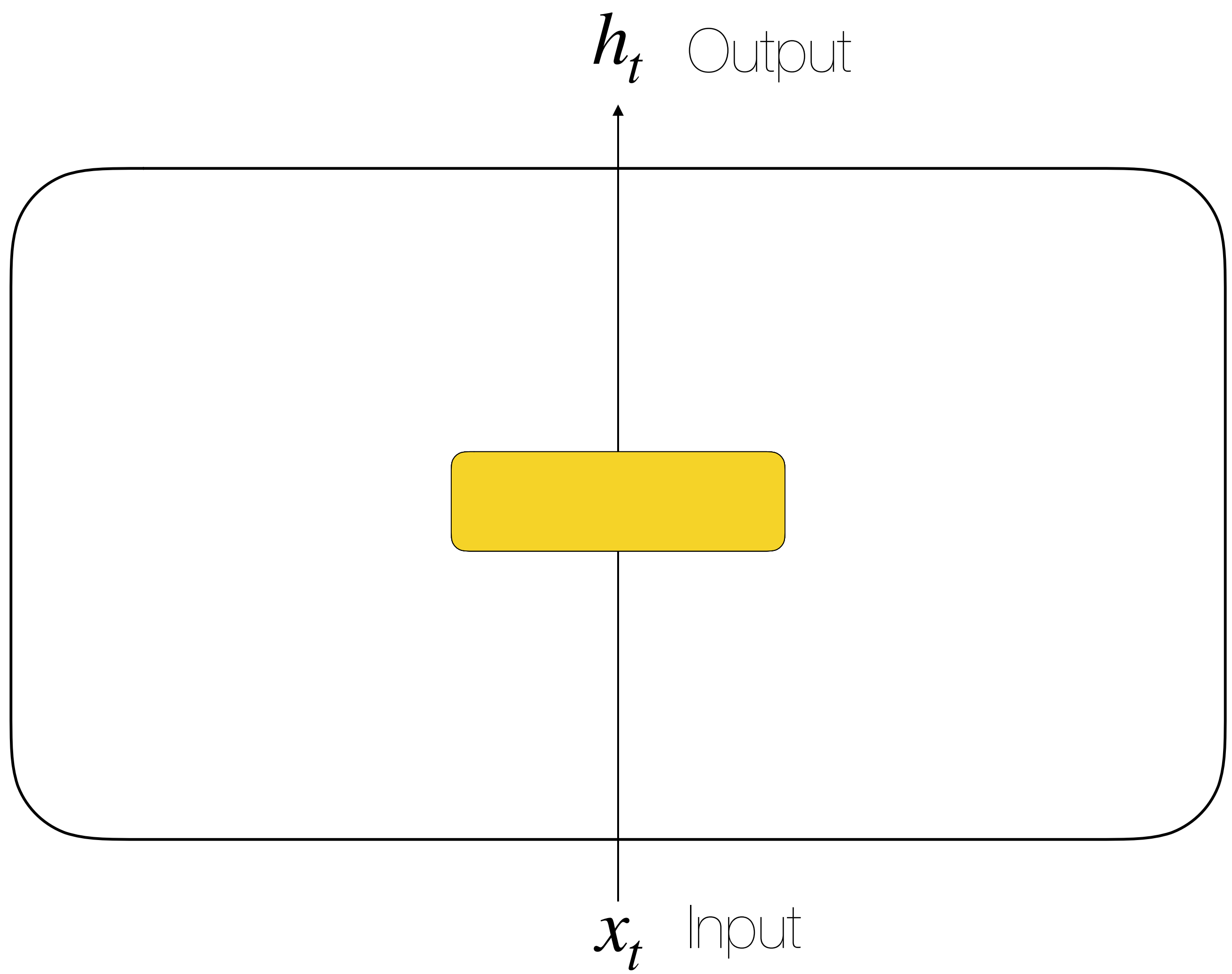
MACHINE

ALGORITHMS
TECHNIQUES
MODELS

OUTPUT TERMS

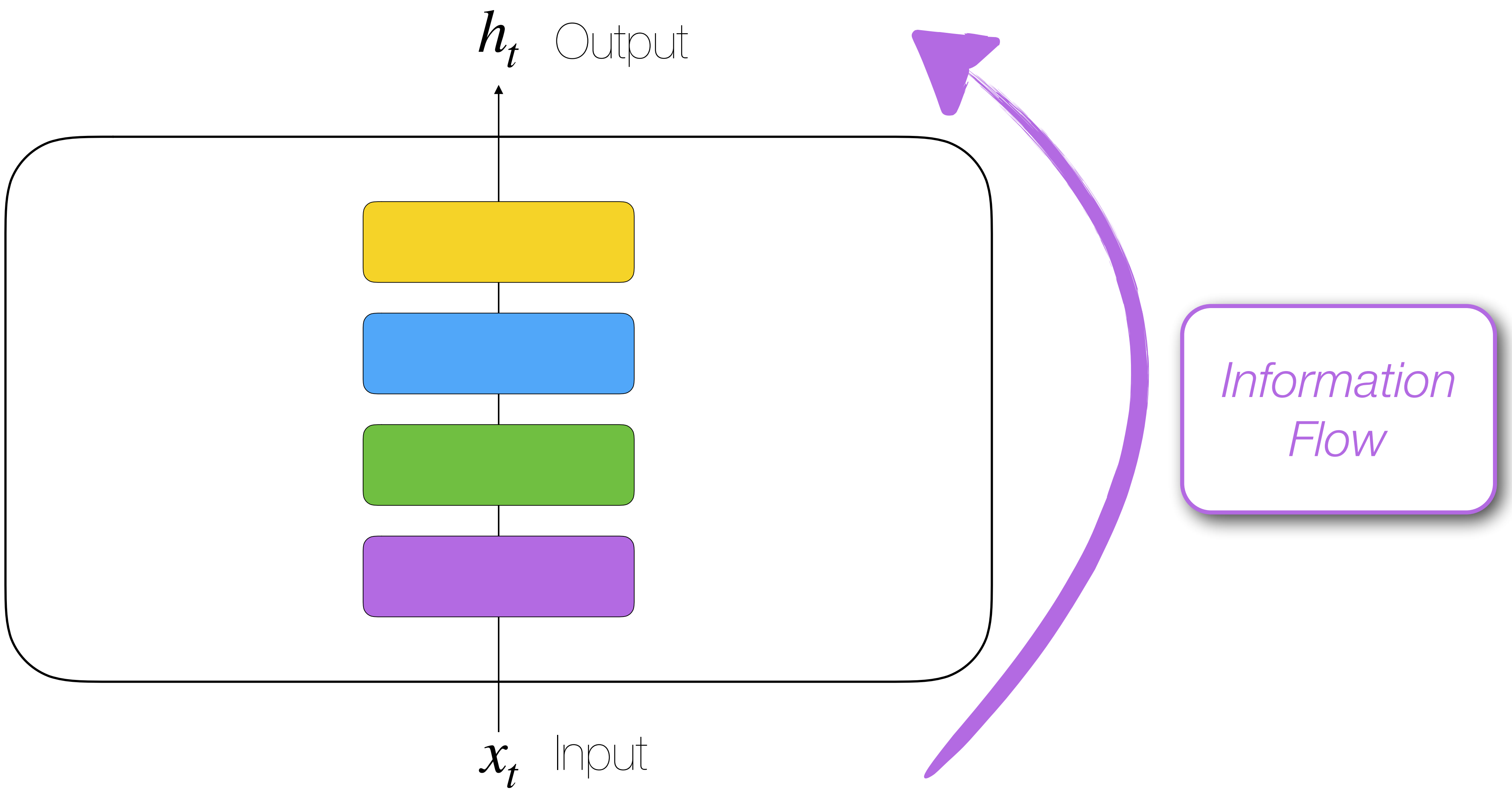
CLASSES
RESPONSES
TARGETS
DEPENDANT VARIABLES

Feed Forward Networks



$$h_t = f\left(x_t\right)$$

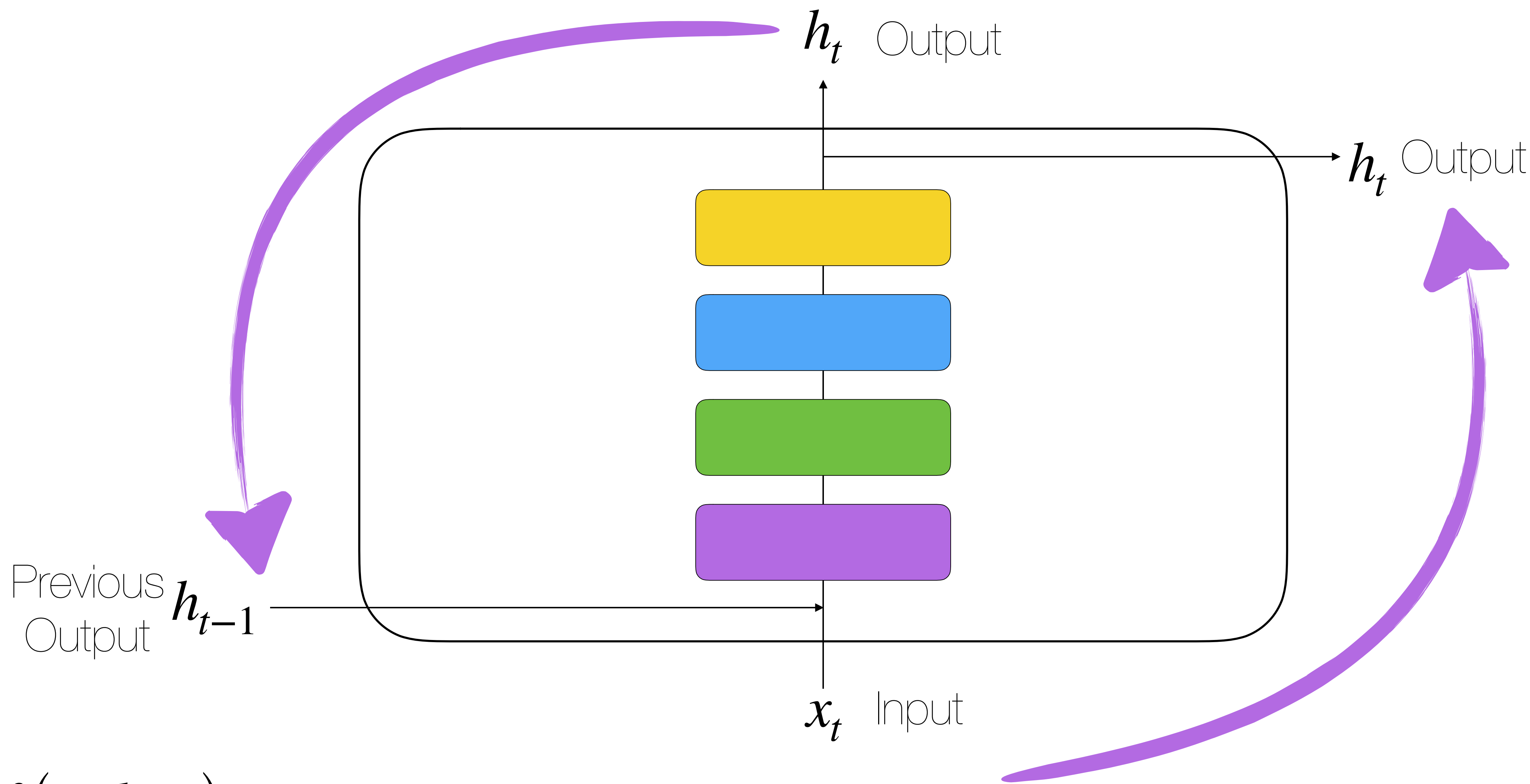
Feed Forward Networks



$$h_t = f(x_t)$$

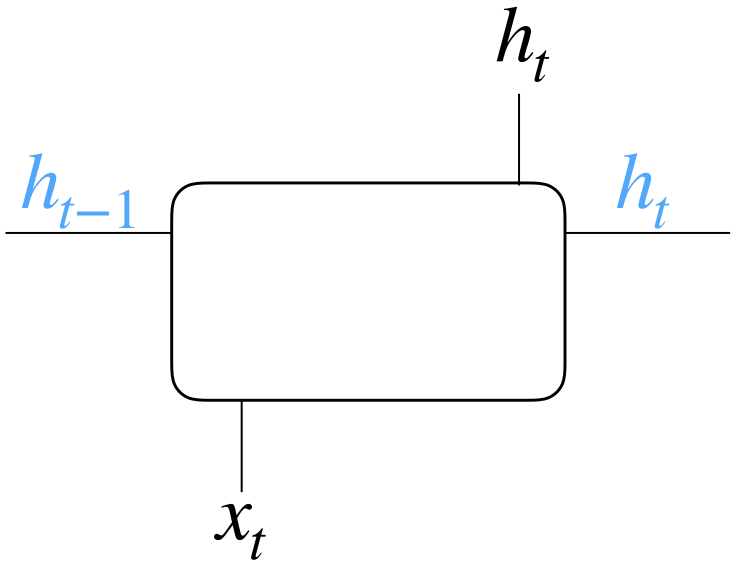
Recurrent Neural Network (RNN)

Information Flow



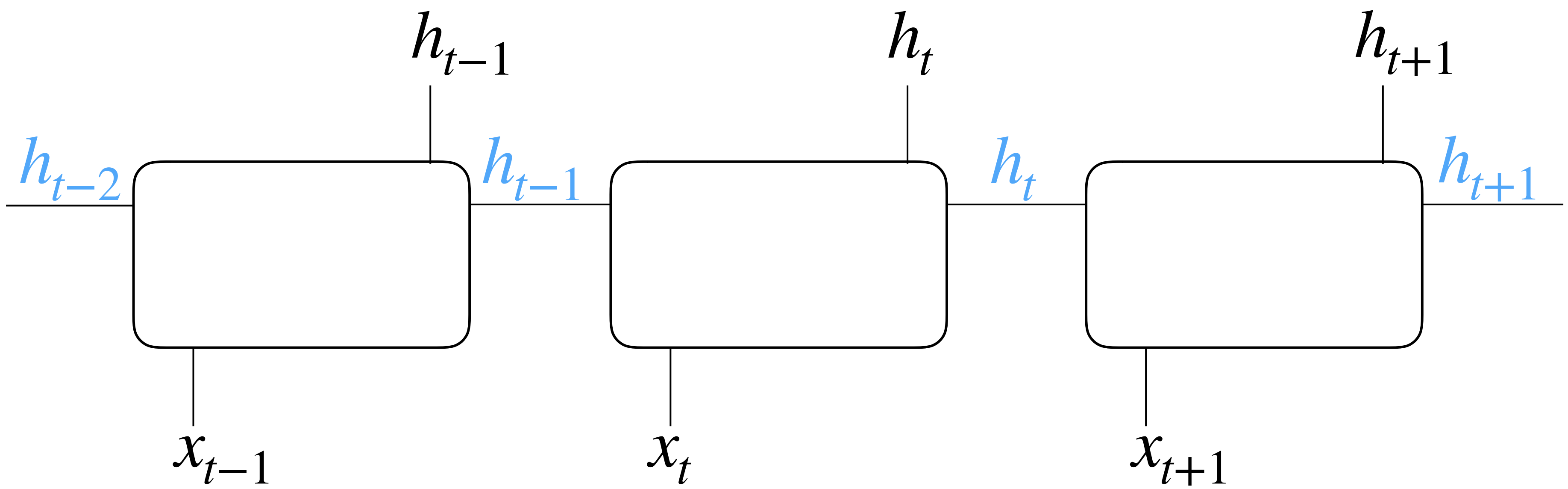
$$h_t = f\left(x_t, h_{t-1}\right)$$

Recurrent Neural Network (RNN)

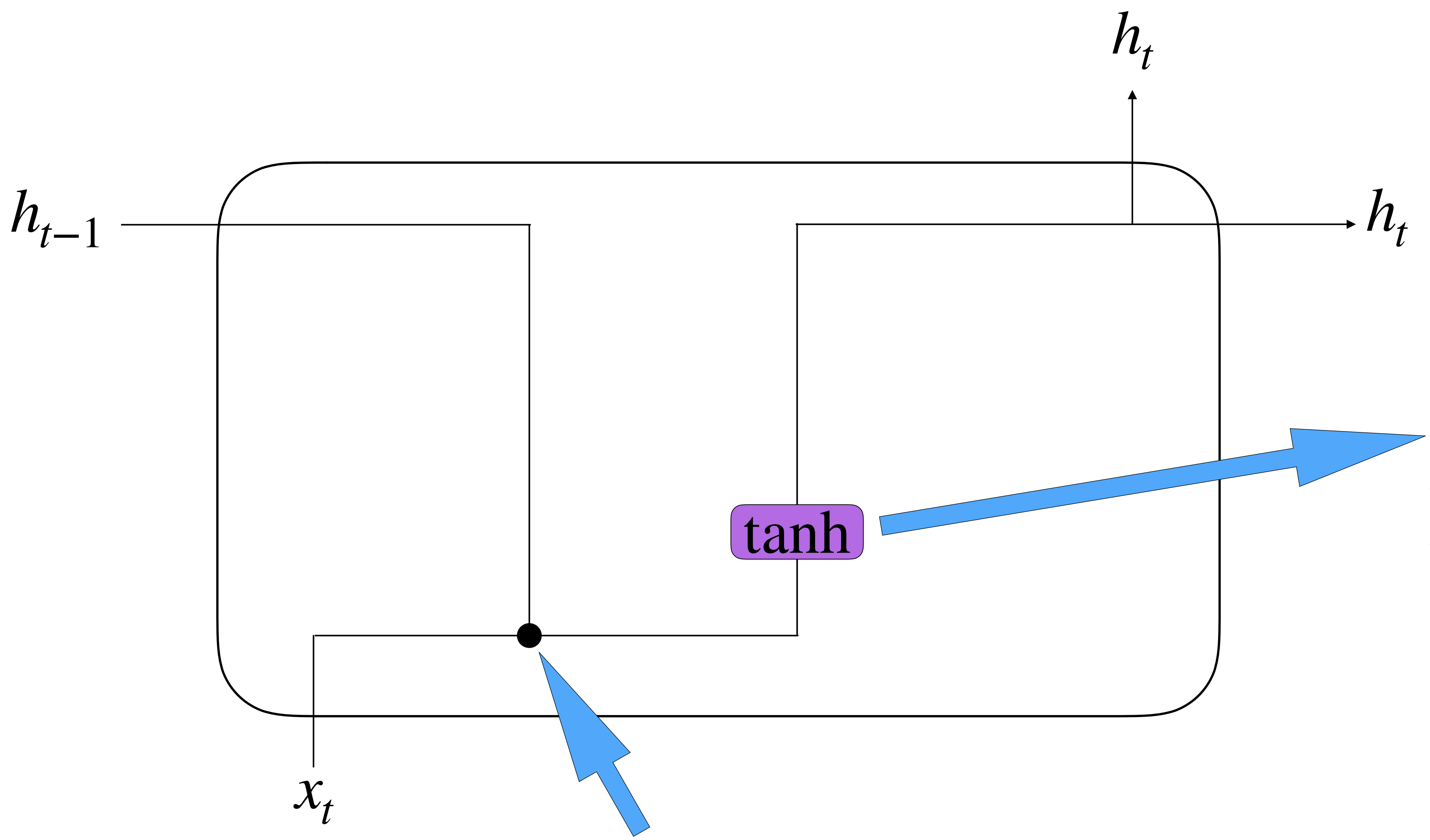


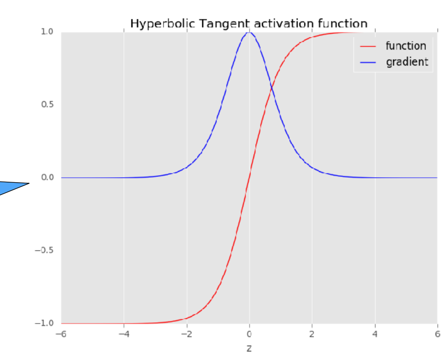
Recurrent Neural Network (RNN)

- Each output depends (implicitly) on all previous outputs.
- Input sequences generate output sequences (seq2seq)



Recurrent Neural Network (RNN)

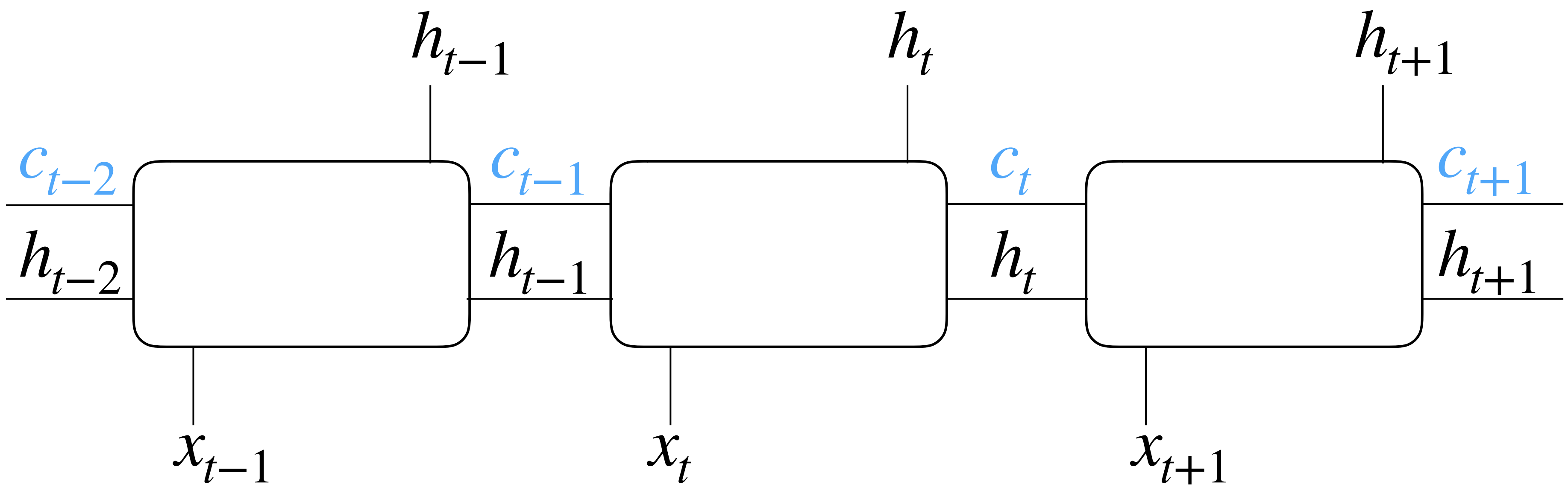




Concatenate both inputs.

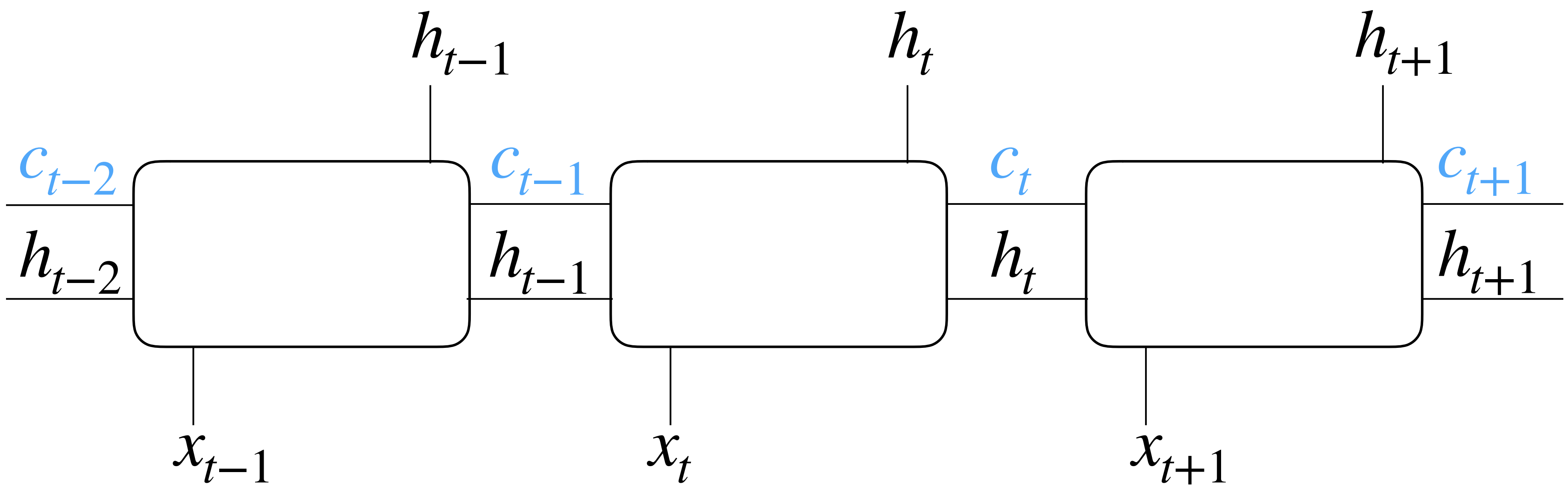
$$h_t = \tanh\left(Wh_{t-1} + Ux_t\right)$$

Long-Short Term Memory (LSTM)



Long-Short Term Memory (LSTM)

- What if we want to keep explicit information about previous states (memory)?
- How much information is kept, can be controlled through gates.
- LSTMs were first introduced in 1997 by Hochreiter and Schmidhuber

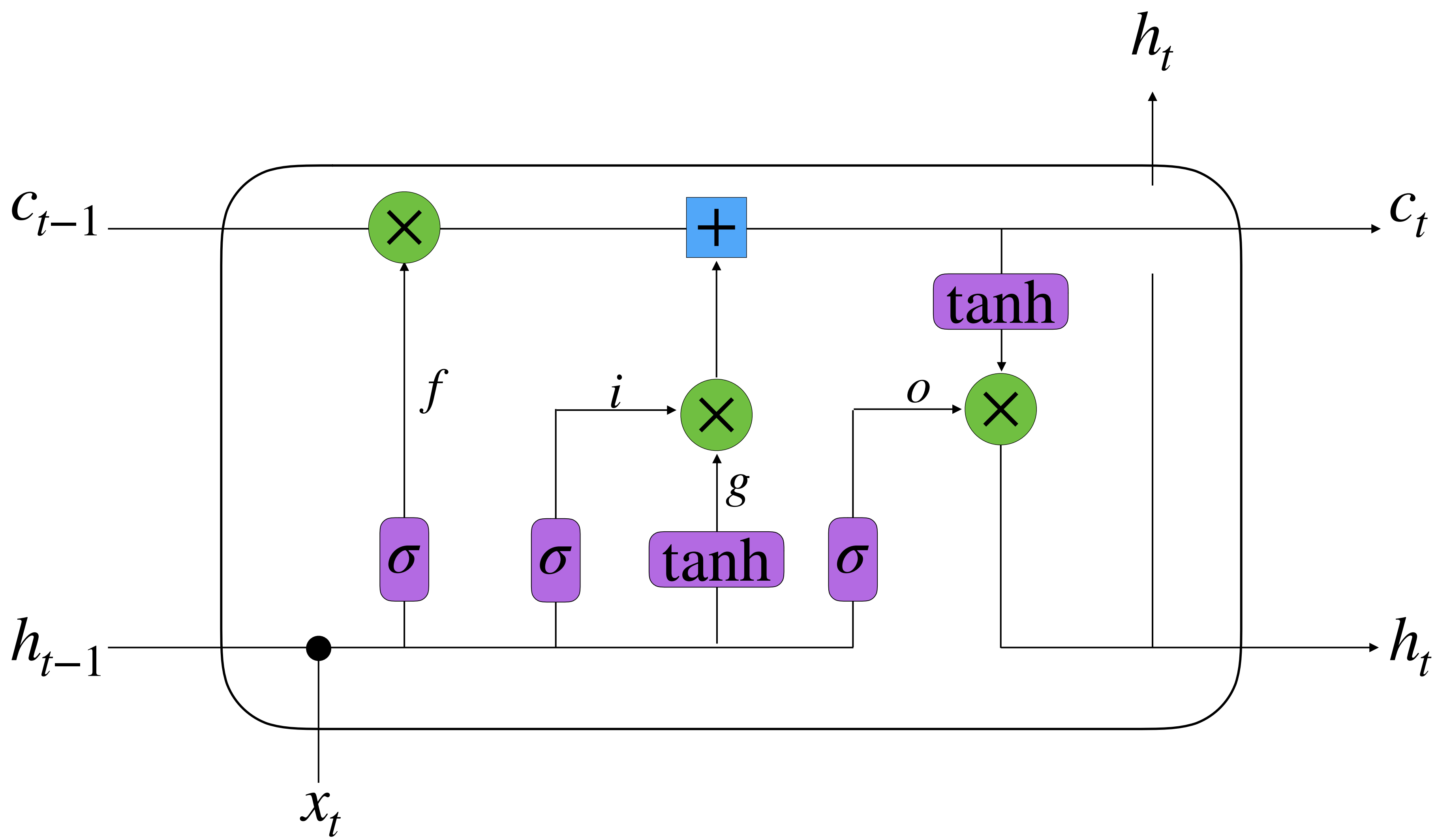


Long-Short Term Memory (LSTM)

Element wise addition

Element wise multiplication

1− 1 minus the input



$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

www.bgoncalves.com

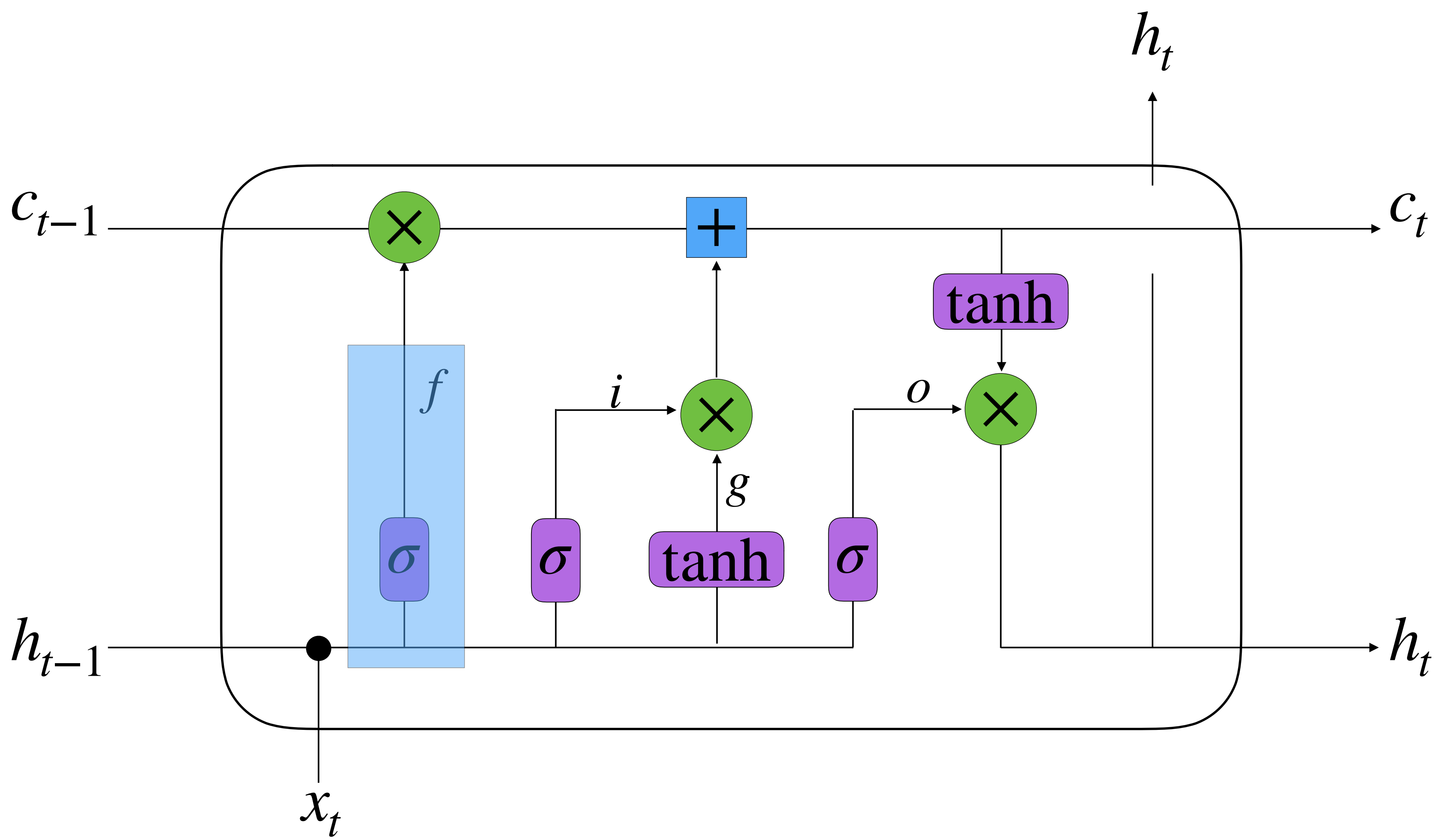
Long-Short Term Memory (LSTM)

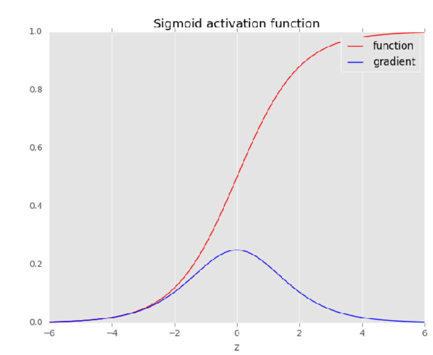
+ Element wise addition

× Element wise multiplication

1− 1 minus the input

Forget gate: How much of the previous state should be kept?





$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

www.bgoncalves.com

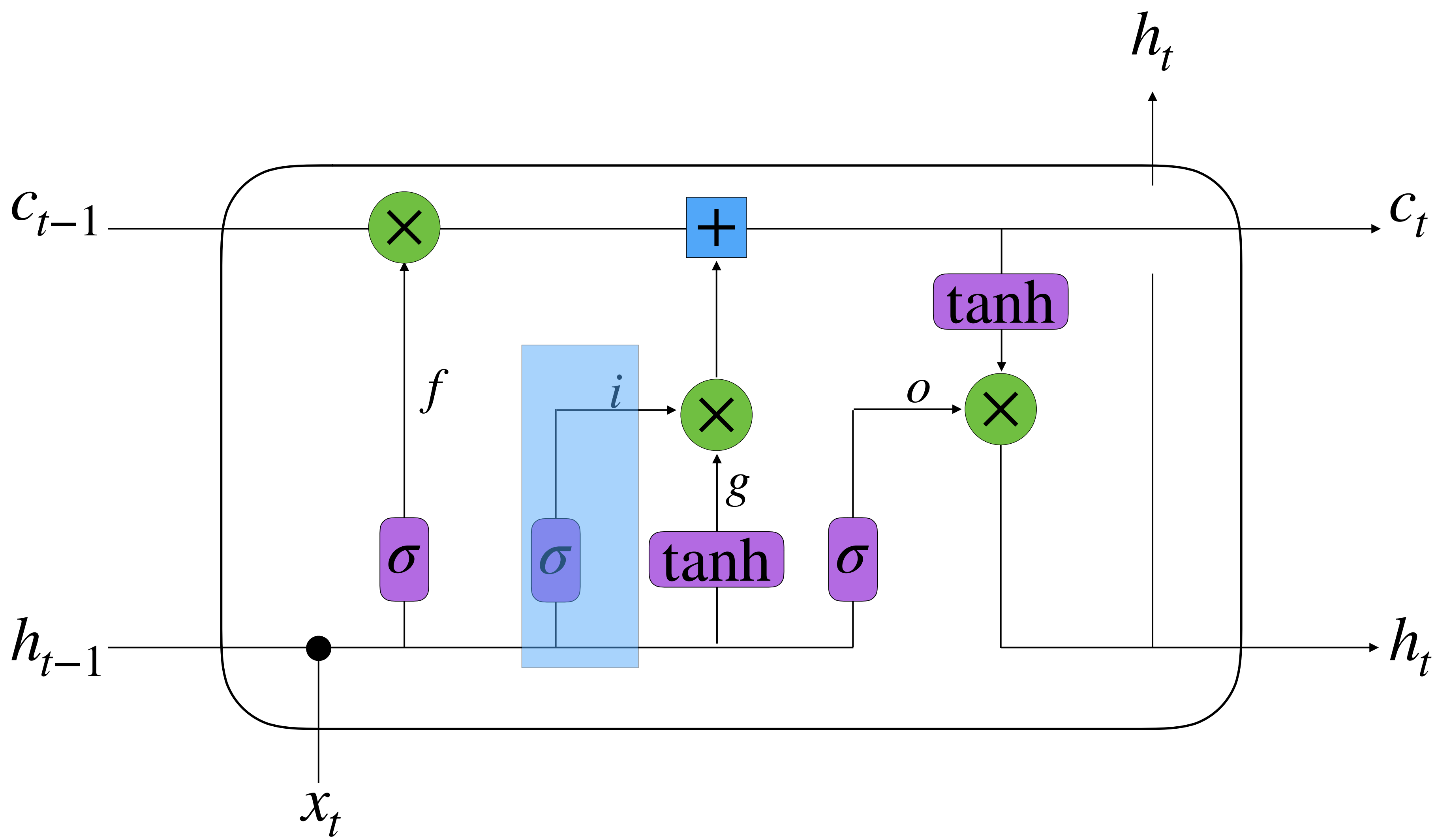
Long-Short Term Memory (LSTM)

Element wise addition

Element wise multiplication

1− 1 minus the input

Input gate: How much of the previous output should be remembered?



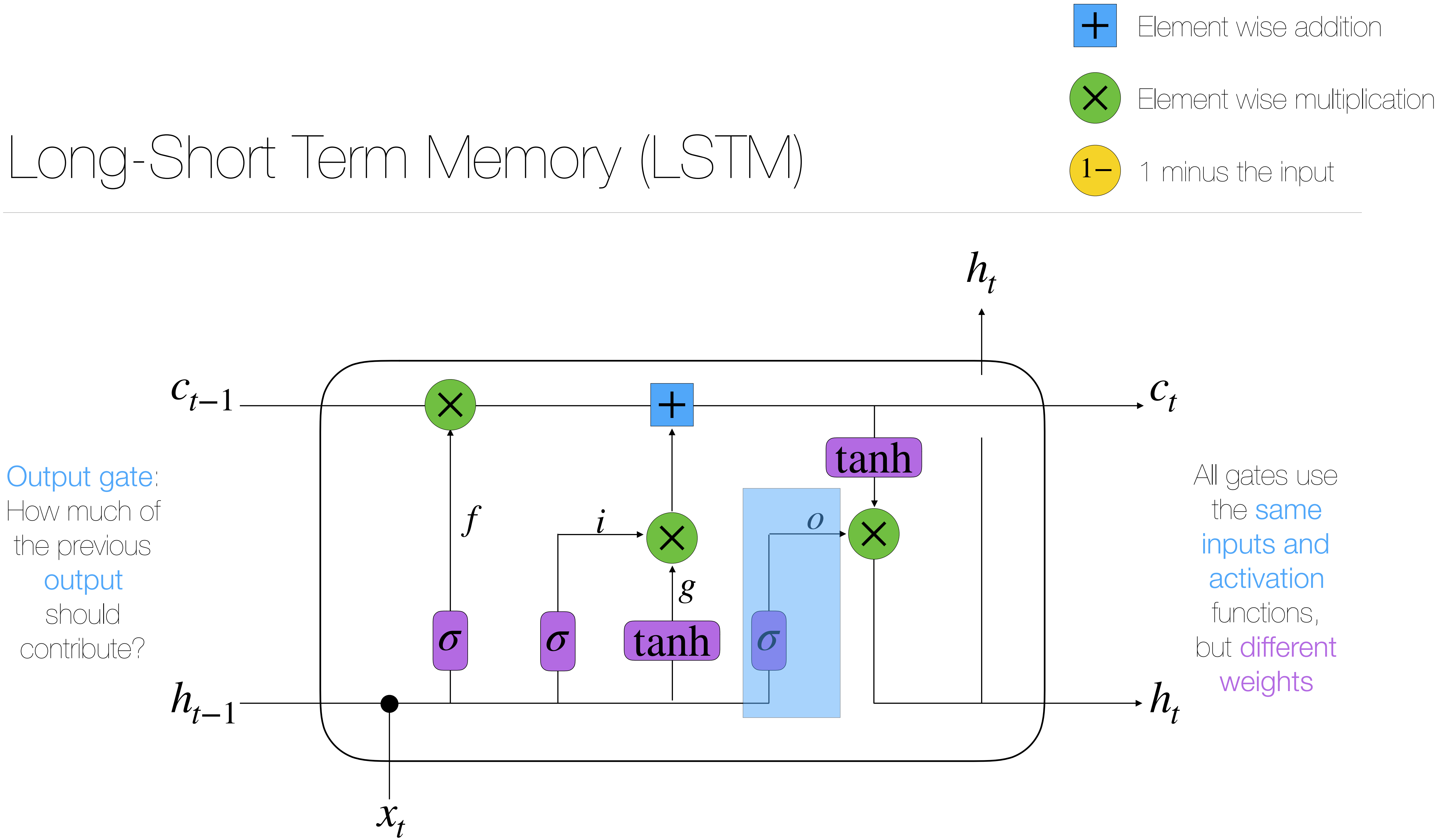
$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

www.bgoncalves.com

Long-Short Term Memory (LSTM)



$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

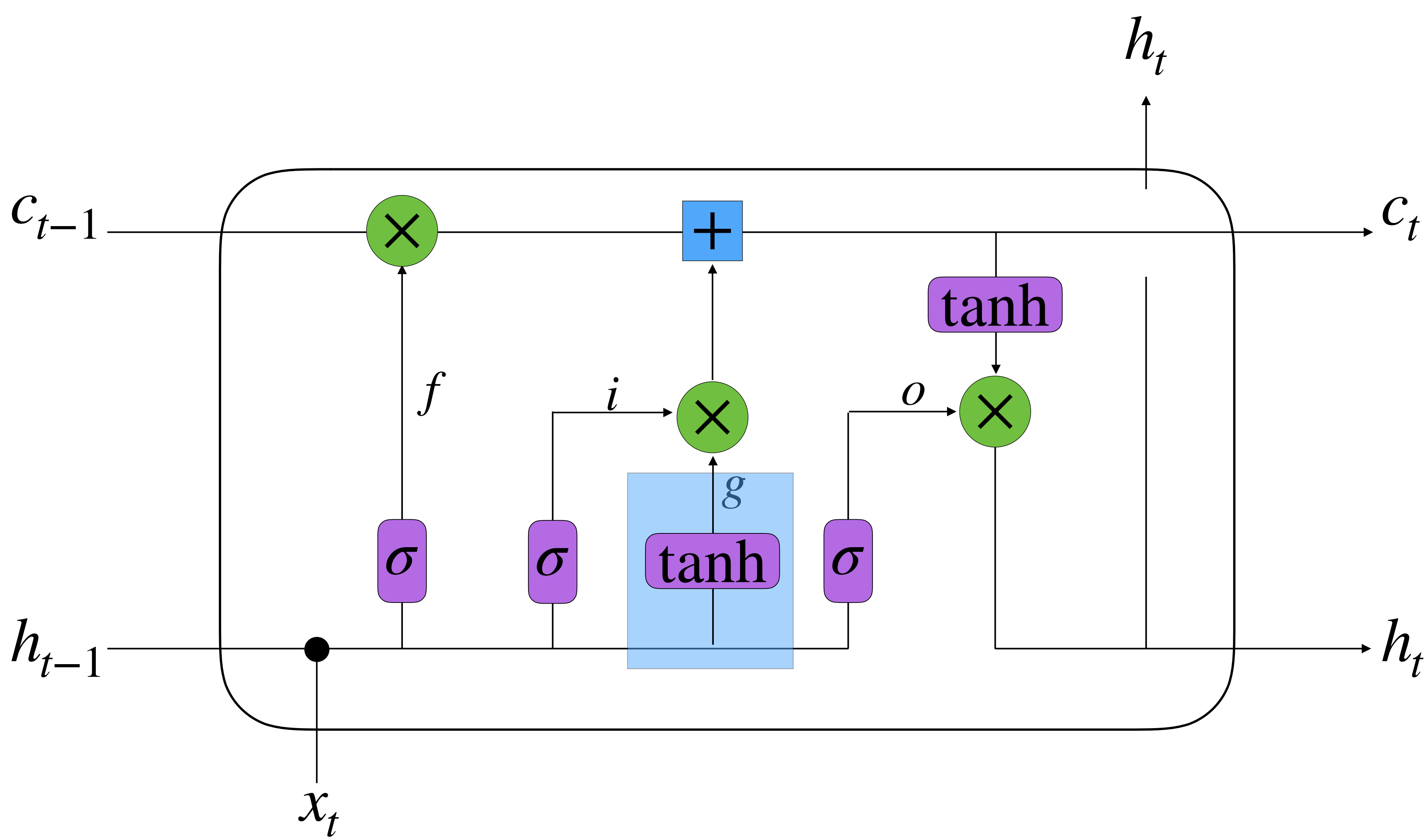
www.bgoncalves.com

Long-Short Term Memory (LSTM)

Element wise addition

Element wise multiplication

1 minus the input



Output gate: How much of the previous output should contribute?

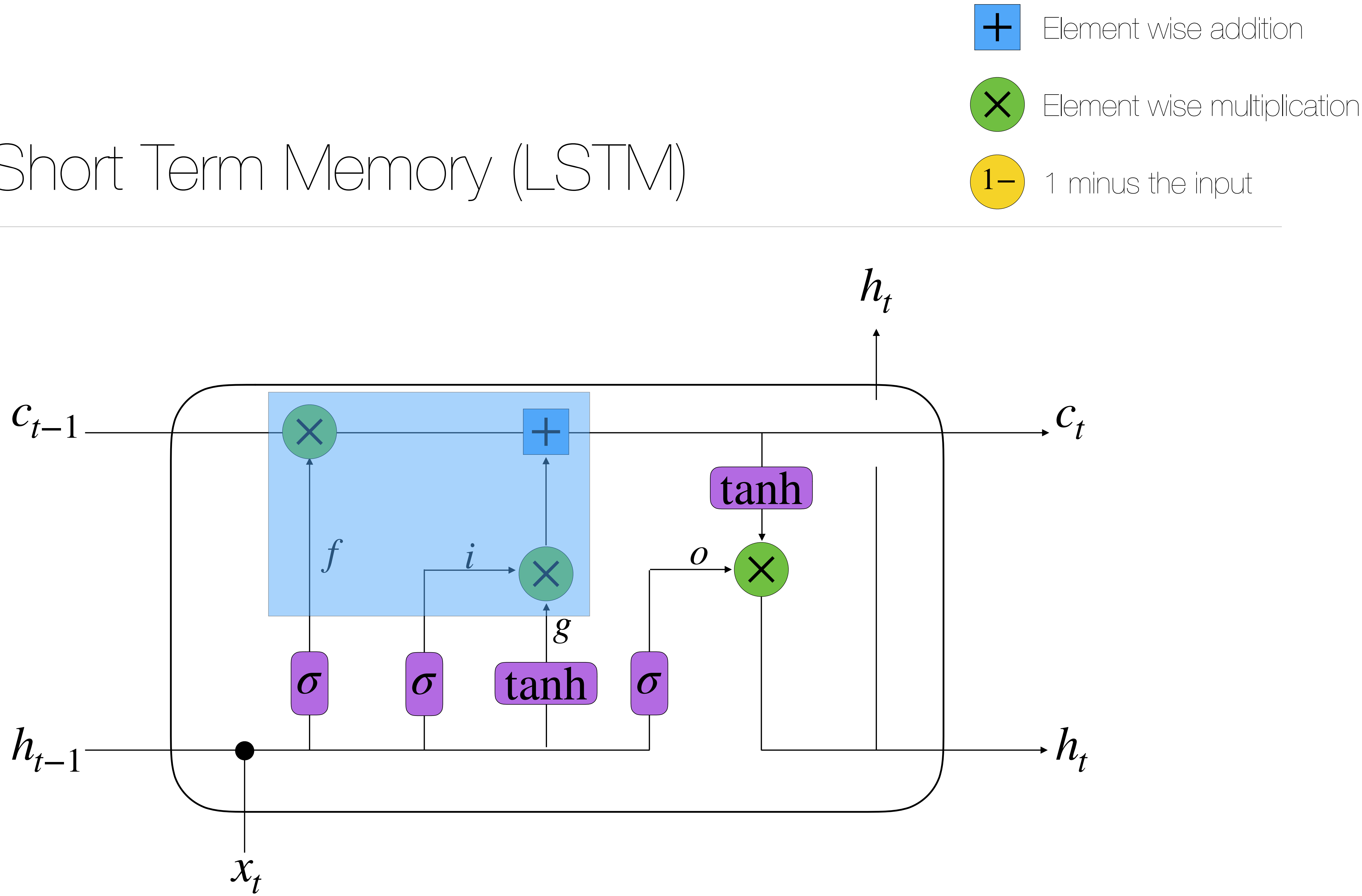
$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

www.bgoncalves.com

Long-Short Term Memory (LSTM)



State:
Update the
current state

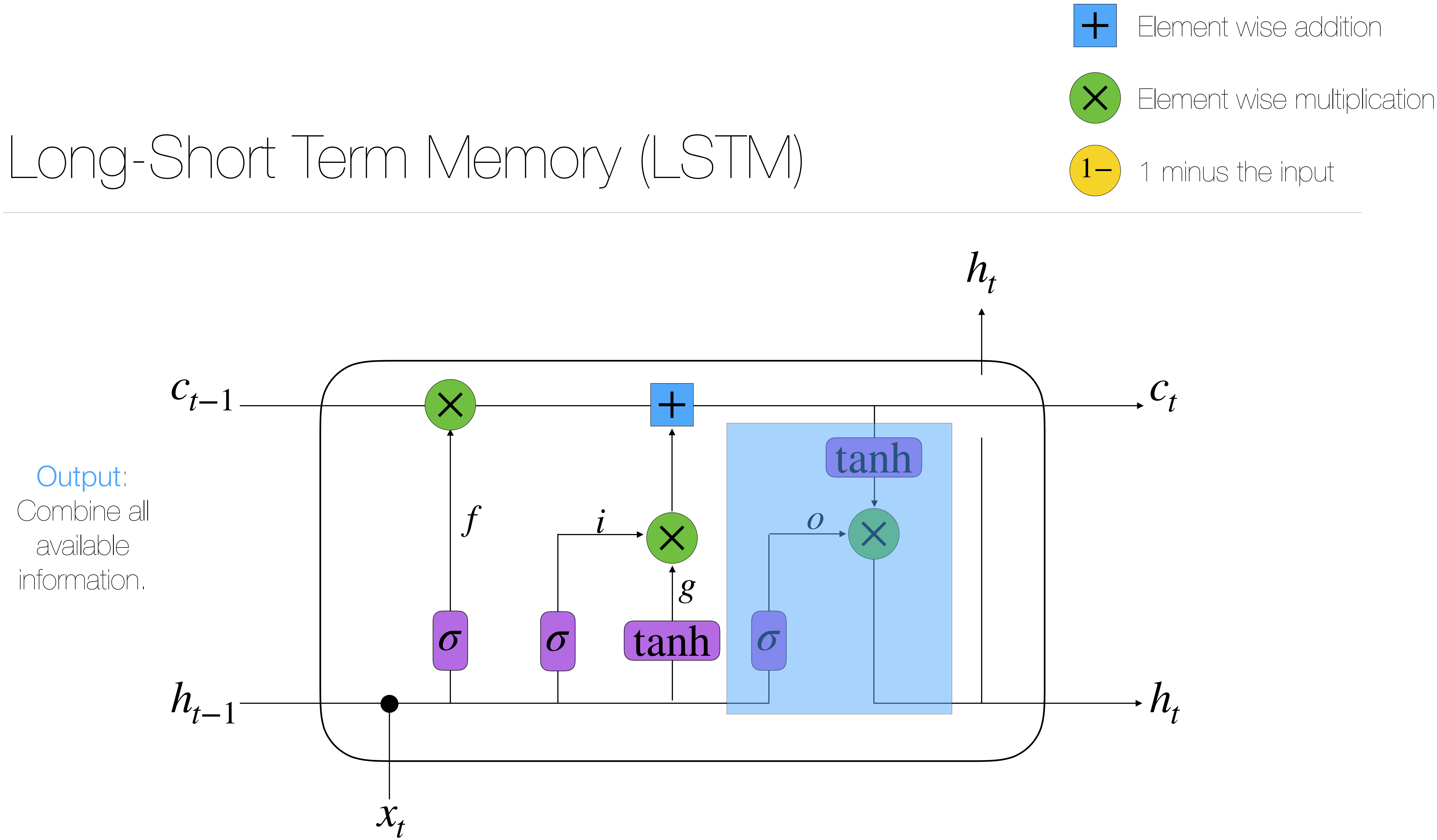
$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

www.bgoncalves.com

Long-Short Term Memory (LSTM)



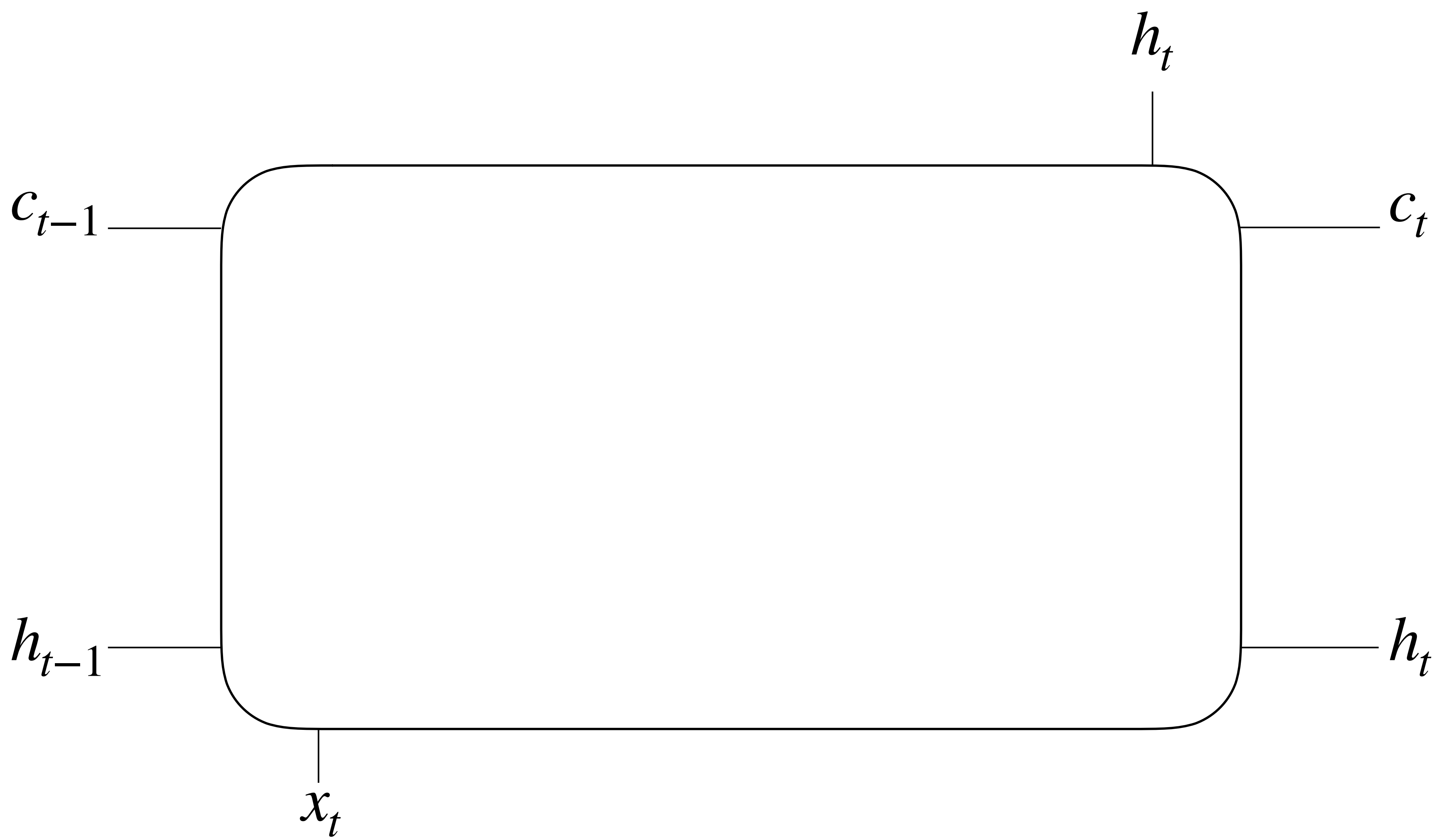
$$f = \sigma\left(W_f h_{t-1} + U_f x_t\right) \qquad g = \tanh\left(W_g h_{t-1} + U_g x_t\right)$$

$$i = \sigma\left(W_i h_{t-1} + U_i x_t\right) \qquad c_t = \left(c_{t-1} \otimes f\right) + \left(g \otimes i\right)$$

$$o = \sigma\left(W_o h_{t-1} + U_o x_t\right) \qquad h_t = \tanh\left(c_t\right) \otimes o$$

www.bgoncalves.com

Neural Networks?



Or legos?

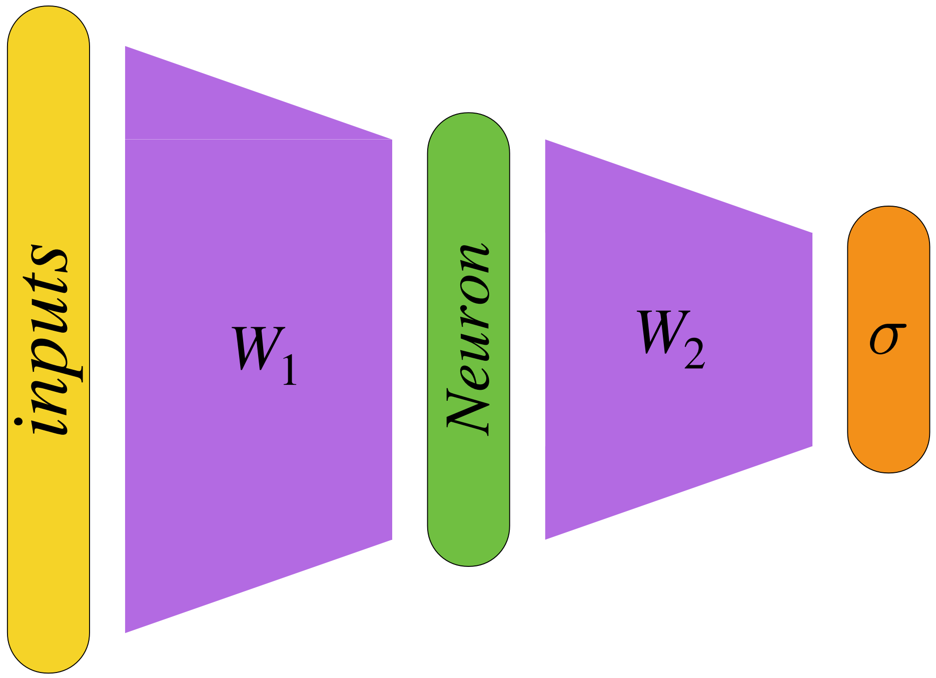
https://keras.io

Keras

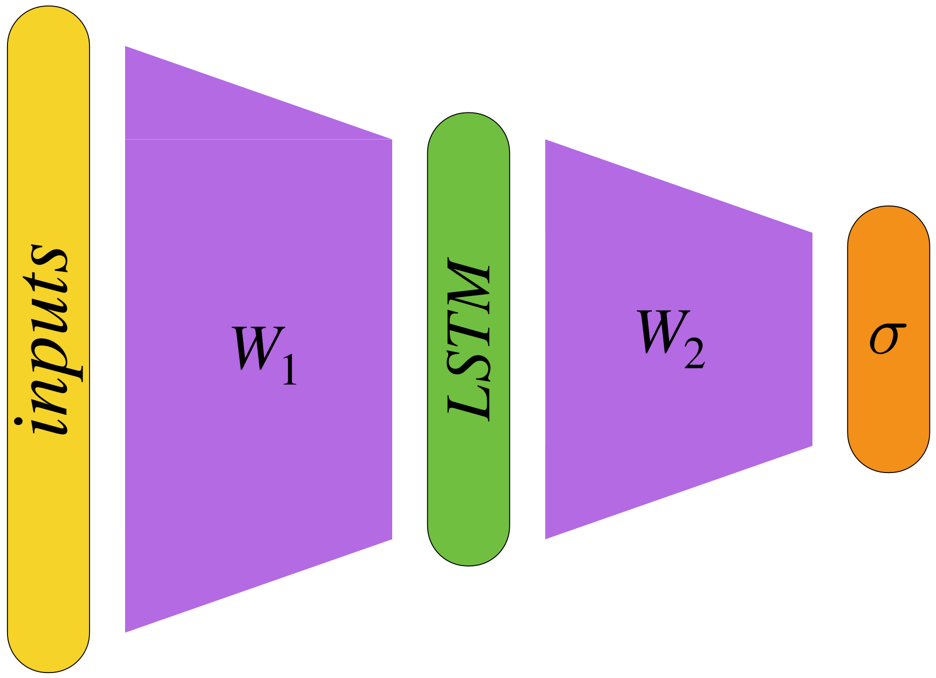
https://keras.io

- Open Source neural network library written in Python
- TensorFlow, Microsoft Cognitive Toolkit or Theano backends
- Enable fast experimentation
- Created and maintained by François Chollet, a Google engineer.
- Implements Layers, Objective/Loss functions, Activation functions, Optimizers, etc…

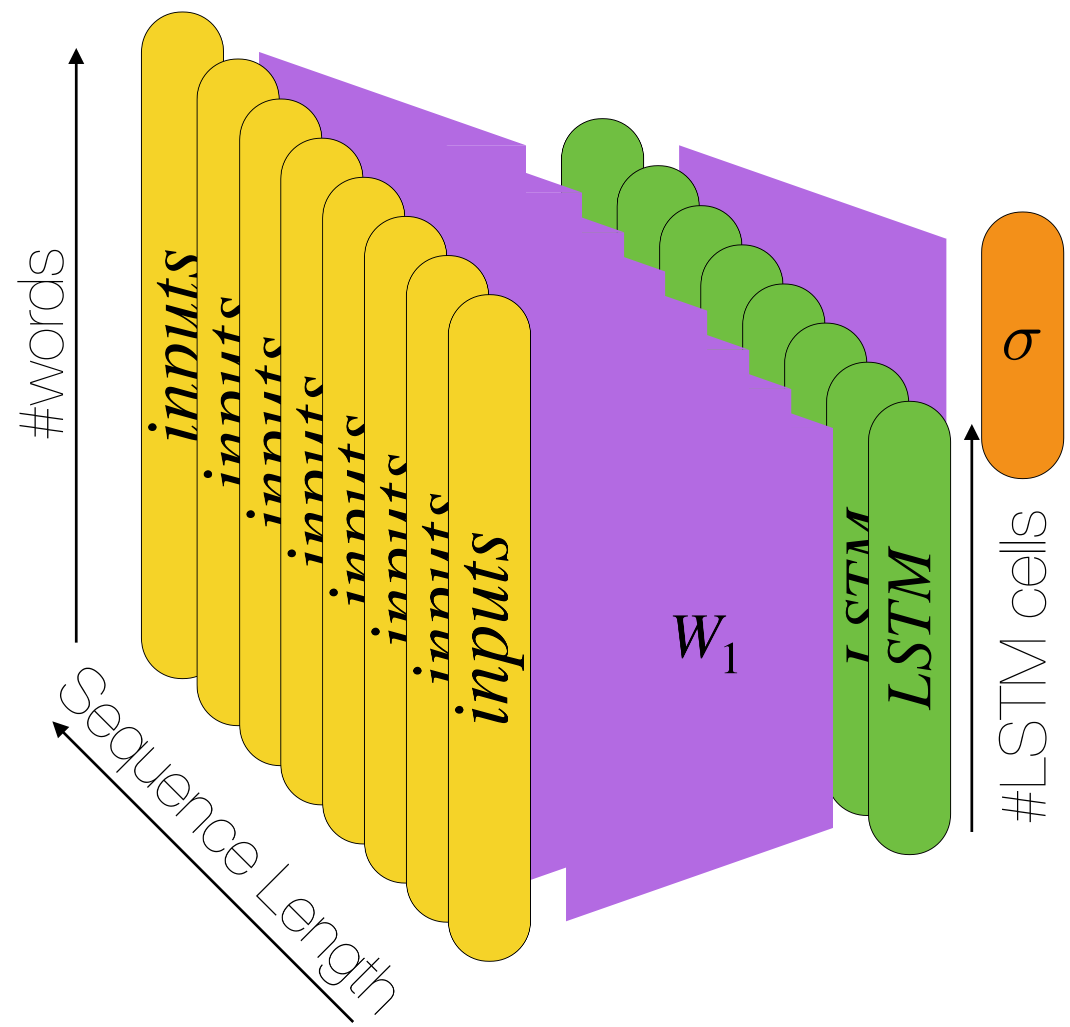
Using LSTMs



Using LSTMs



Using LSTMs



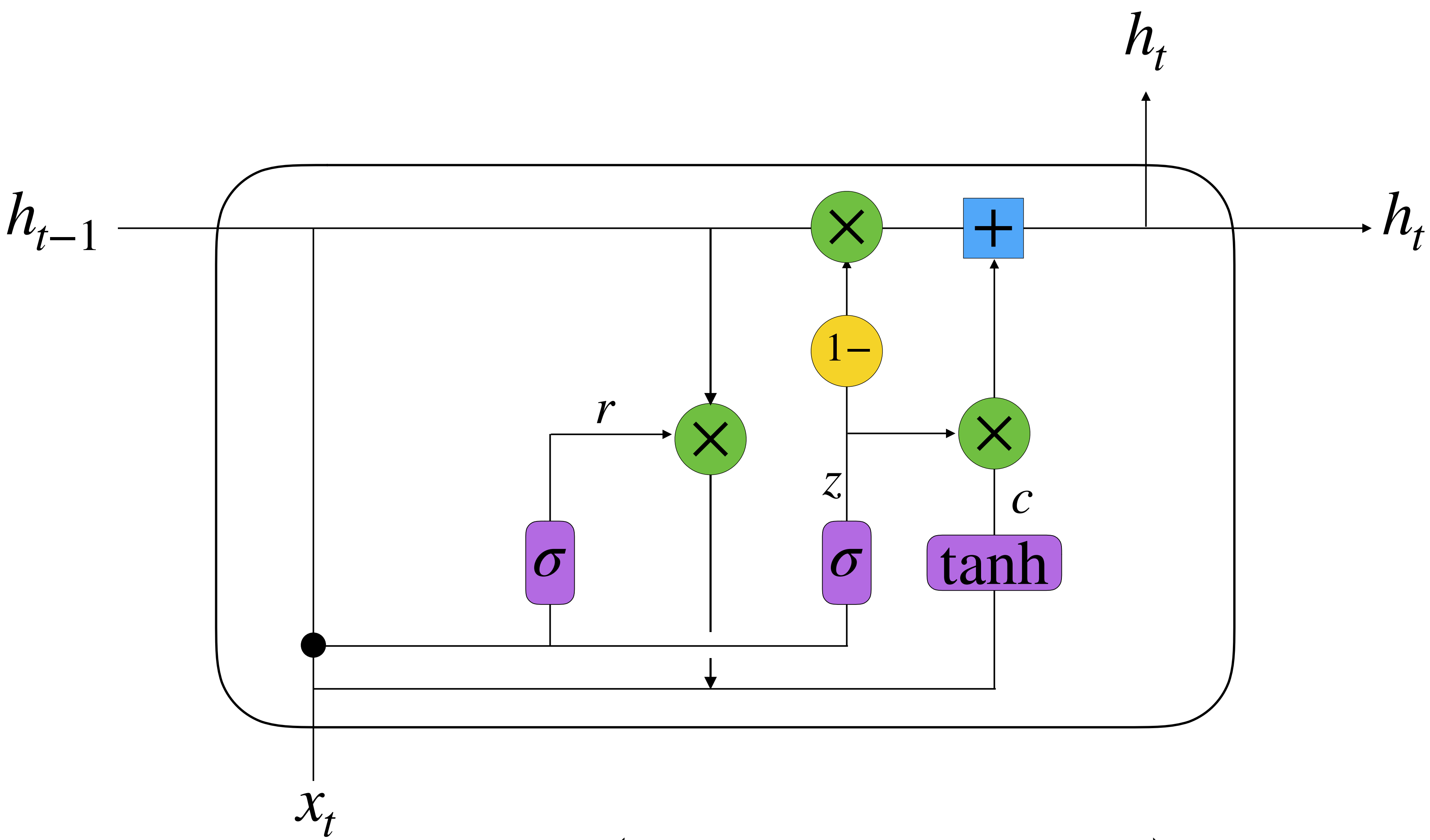
Applications

- Language Modeling and Prediction
- Speech Recognition
- Machine Translation
- Part-of-Speech Tagging
- Sentiment Analysis
- Summarization
- Time series forecasting

Gated Recurrent Unit (GRU)

- Introduced in 2014 by Cho
- Meant to solve the Vanishing Gradient Problem
- Can be considered as a simplification of LSTMs
- Similar performance to LSTM in some applications, better performance for smaller datasets.

Gated Recurrent Unit (GRU)



$$z = \sigma\left(W_z h_{t-1} + U_z x_t\right)$$

$$r = \sigma\left(W_r h_{t-1} + U_r x_t\right)$$

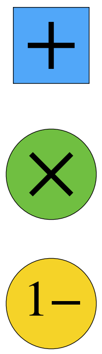
$$c = \tanh\left(W_c\left(h_{t-1} \otimes r\right) + U_c x_t\right)$$

$$h_t = (z \otimes c) + \left((1 - z) \otimes h_{t-1}\right)$$

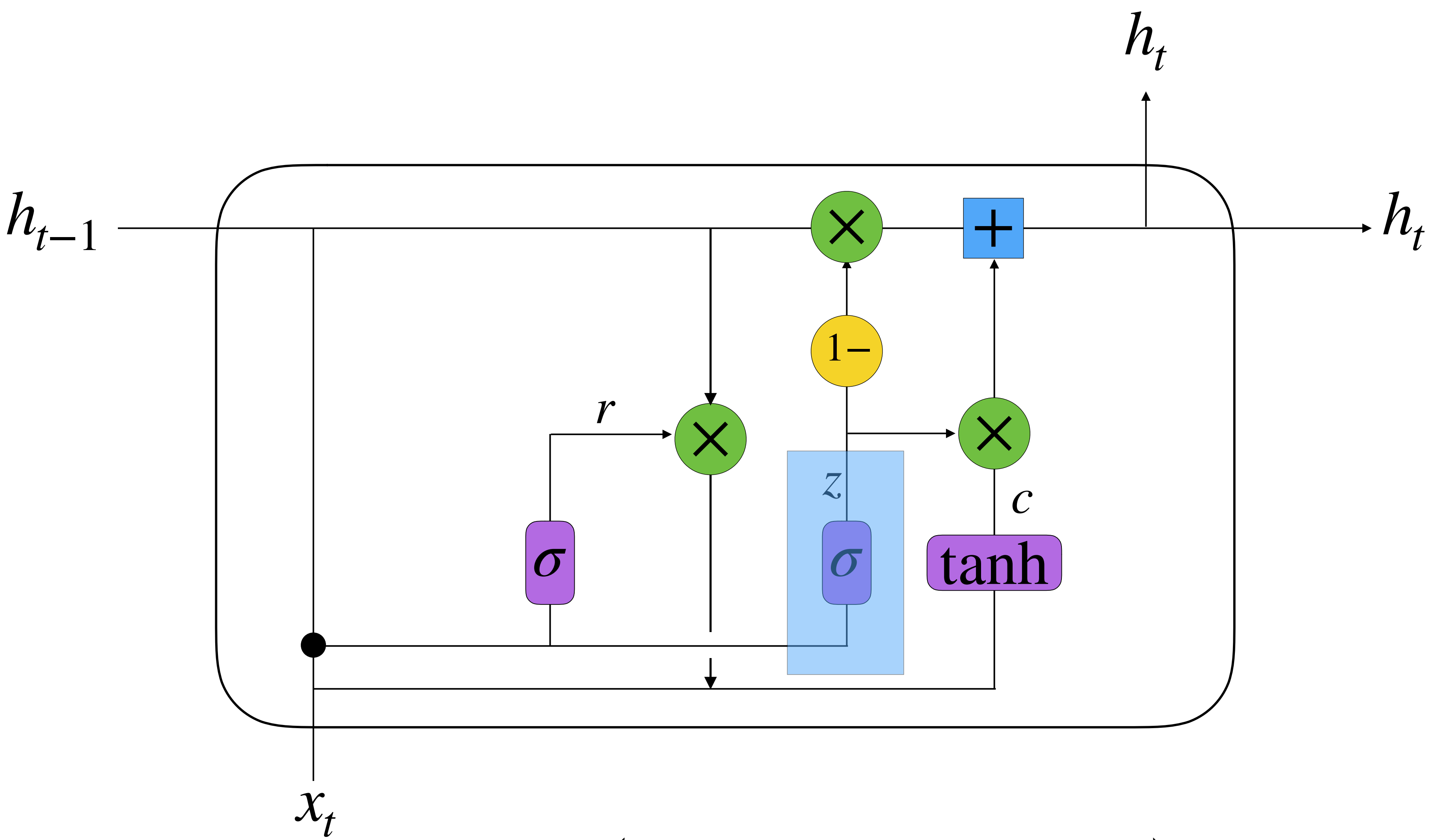
@bgoncalves

www.bgoncalves.com

Gated Recurrent Unit (GRU)



Update gate: How much of the previous state should be kept?



$$z = \sigma\left(W_z h_{t-1} + U_z x_t\right)$$

$$r = \sigma\left(W_r h_{t-1} + U_r x_t\right)$$

$$c = \tanh\left(W_c\left(h_{t-1} \otimes r\right) + U_c x_t\right)$$

$$h_t = (z \otimes c) + \left((1-z) \otimes h_{t-1}\right)$$

@bgoncalves

www.bgoncalves.com

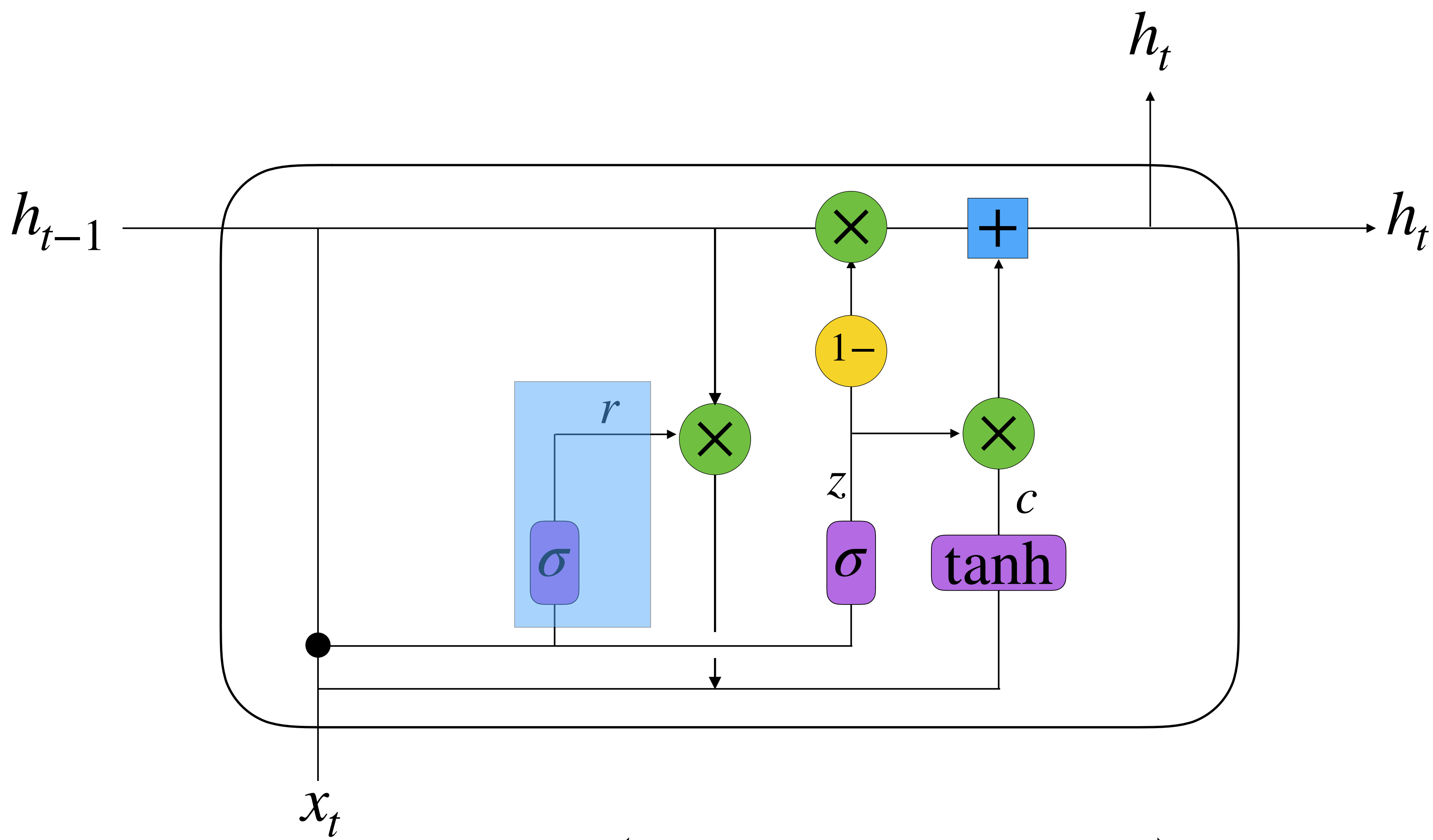
1− 1 minus the input

Element wise addition

Element wise multiplication

Gated Recurrent Unit (GRU)

Reset gate:
How much of
the previous
output should
be removed?



$$c = \tanh\left(W_c\left(h_{t-1} \otimes r\right) + U_c x_t\right)$$

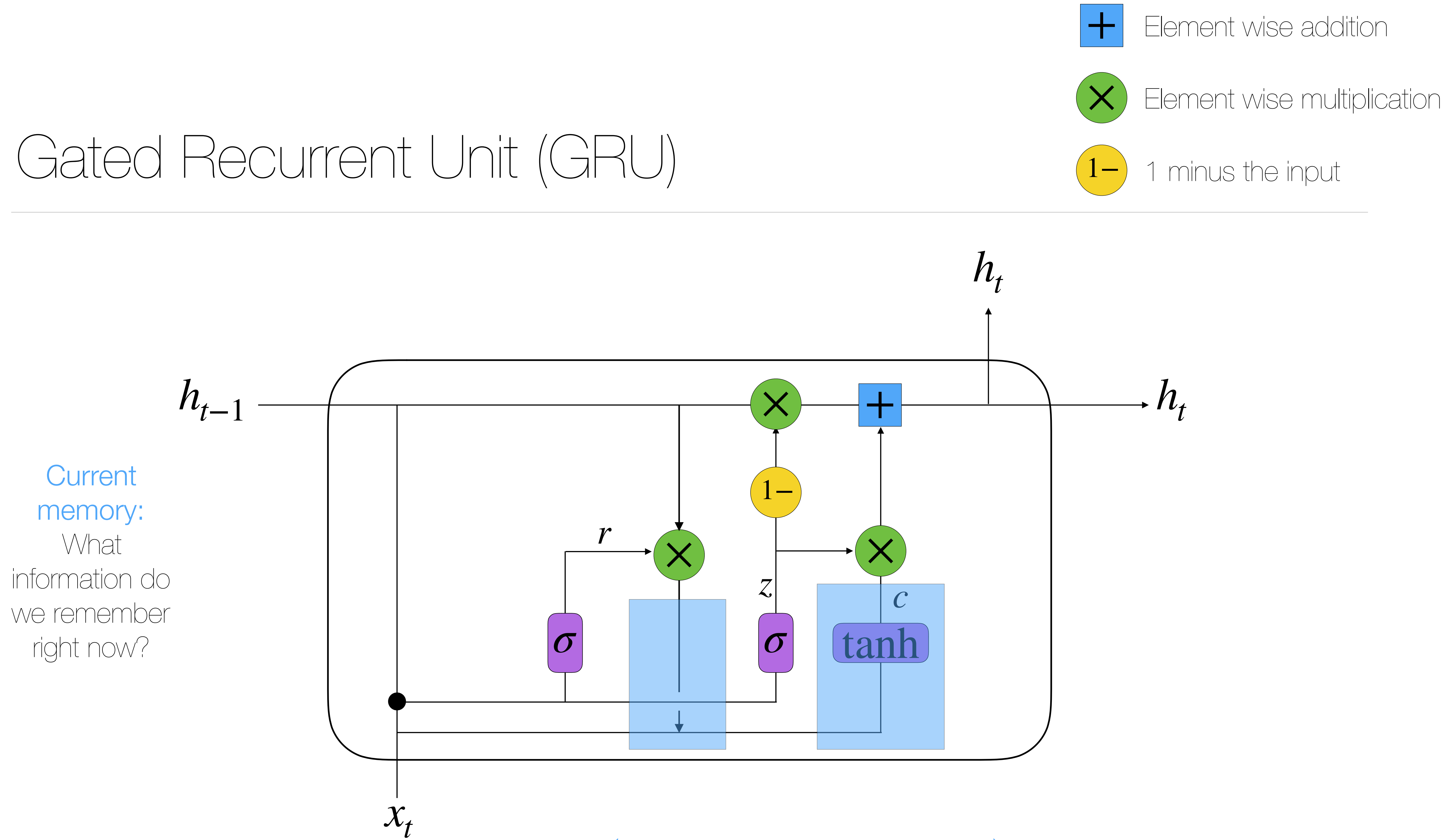
$$z = \sigma\left(W_z h_{t-1} + U_z x_t\right)$$

$$r = \sigma\left(W_r h_{t-1} + U_r x_t\right)$$

$$h_t = (z \otimes c) + \left((1 - z) \otimes h_{t-1}\right)$$

@bgoncalves www.bgoncalves.com

Gated Recurrent Unit (GRU)



$$c = \tanh\left(W_c\left(h_{t-1} \otimes r\right) + U_c x_t\right)$$

$$z = \sigma\left(W_z h_{t-1} + U_z x_t\right)$$

$$r = \sigma\left(W_r h_{t-1} + U_r x_t\right)$$

$$h_t = (z \otimes c) + \left((1 - z) \otimes h_{t-1}\right)$$

@bgoncalves

www.bgoncalves.com

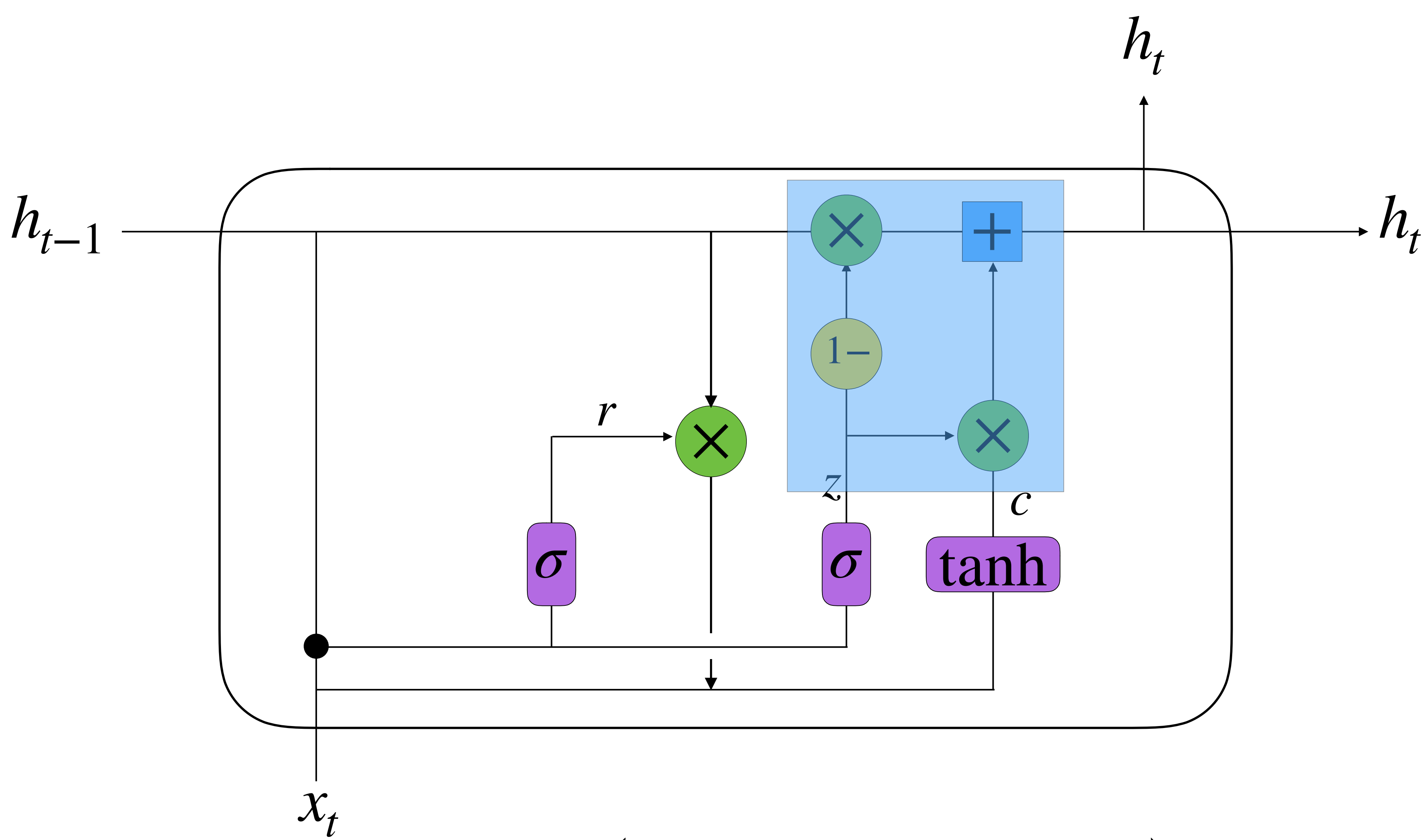
Element wise addition

Element wise multiplication

1 minus the input

Gated Recurrent Unit (GRU)

Output:
Combine all
available
information.



$$z = \sigma\left(W_z h_{t-1} + U_z x_t\right)$$

$$r = \sigma\left(W_r h_{t-1} + U_r x_t\right)$$

$$c = \tanh\left(W_c\left(h_{t-1} \otimes r\right) + U_c x_t\right)$$

$$h_t = (z \otimes c) + \left((1 - z) \otimes h_{t-1}\right)$$