’Pattern Recognition in Audio-Signals’

04 – Models for sequential data II

Kilian Schuster

Hochschule Luzern

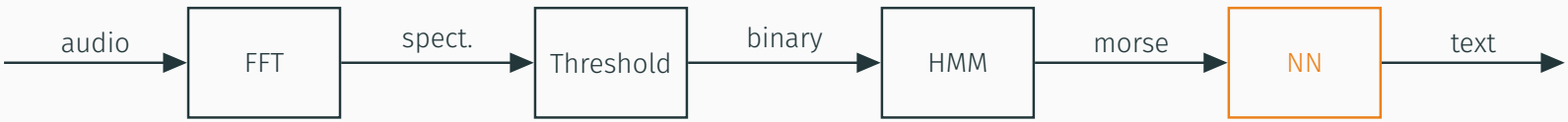
Overview

1. Problem (Example)
2. BNN (Biological Neural Network)
3. ANN (Artificial Neural Network)
4. Schemes of Application
5. RNN (Recurrent Neural Network)
6. Code Snippets
7. Addendum
8. References

Problem (Example)

Morse code

- Morse code – Wikipedia
- Step 1: from audio → binary sequence
  - take spectrogram (FFT)
  - select component at specific frequency (e.g. 600 Hz)
  - rate against threshold
- Step 2: from binary sequence → Morse code
  - run viterbi decoder on HMM (Hidden Markov Model)
- Step 3: from Morse code → text
  - train NN (Neural Network)
  - take Morse code as input, output text



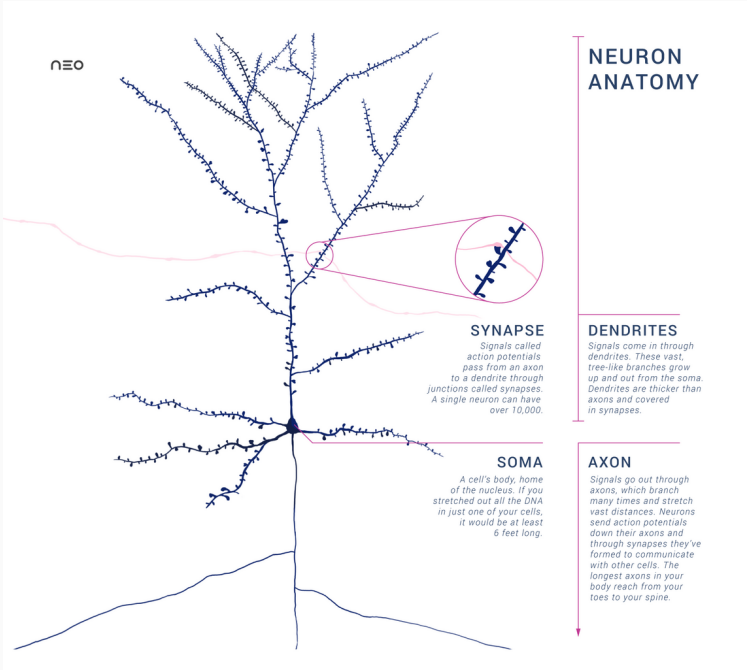
Working hypothesis :

Morse code is not known a priori, only samples (plain & encoded) are given.

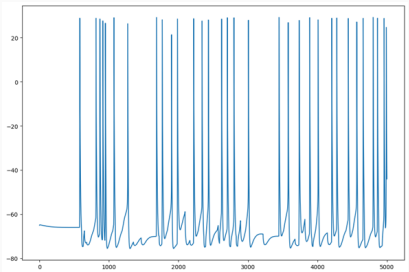
BNN (Biological Neural Network)

Neurons

- Human brain contains about 10–20 billion neurons in the cerebral cortex and 55–70 billion neurons in the cerebellum
- Many different types and topologies of neurons
- Activity represented by electrochemical potential
- Signalling by spikes and firing rate
- Propagation along axon (speed depending on diameter, up to 120 $^{m}/_{s}$)
- Coupling from axon to dendrites via synaptic gap with help of neurotransmitters

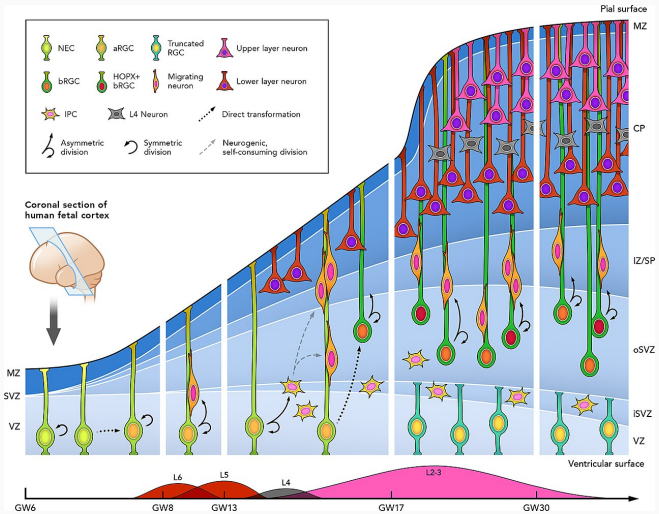


[Source]


Potential wrt. time [Source]

Development & learning

- Most of the neurons fully developed at birth
- Only minor genesis of neurons during lifetime (details still under investigations)
- Maximal density of interconnections at age 2-3 year (preferred age to learn language)
- Learning is related to synaptic adaptations
- Pruning of unused connections
- Hebb's postulate: *'Neurons that fire together, wire together'* (D.O. Hebb: 'The Organization of Behavior')
- Many different forms of synaptic plasticity (very active field of research)



GW : gestational week [Source]

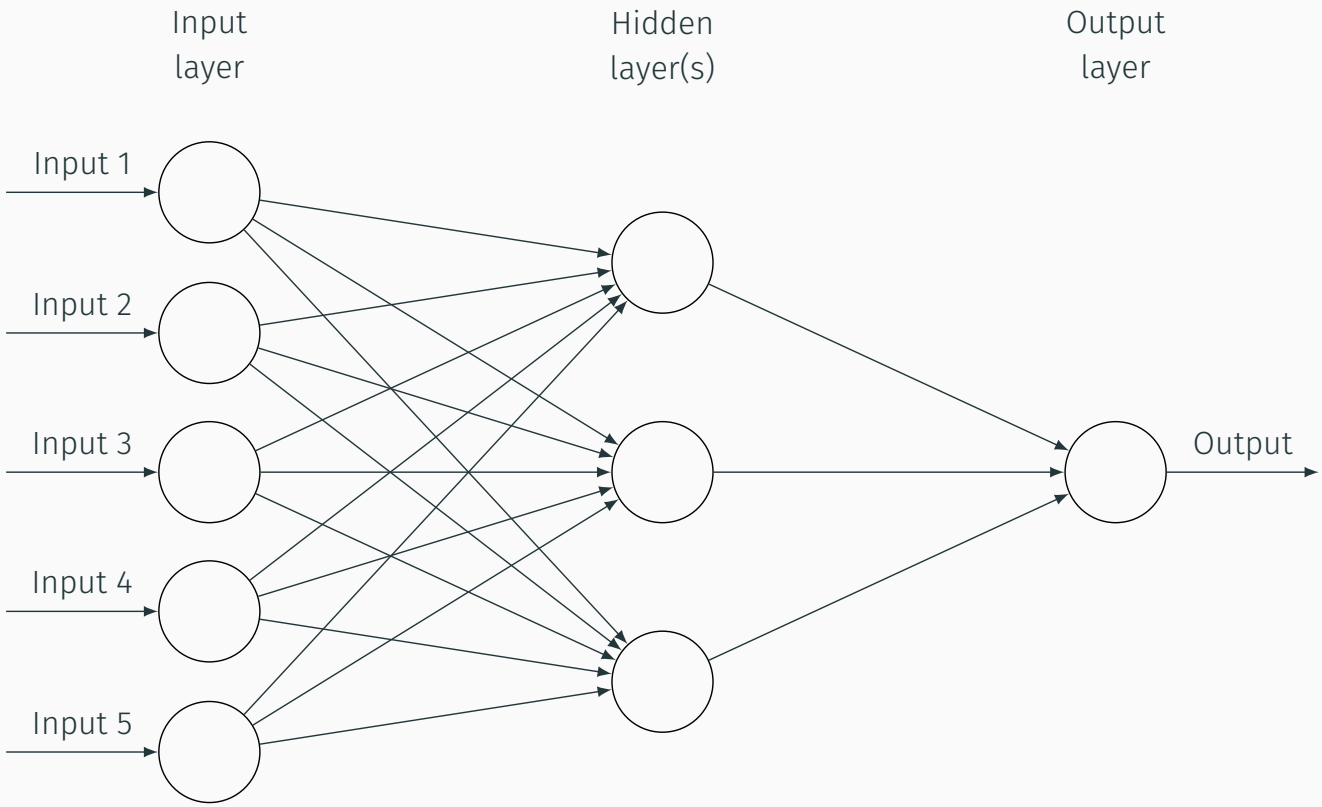
ANN (Artificial Neural Network)

History

- Dawn (1941-1956) First artificial neurons, influenced by Turing's 'On computable numbers', by W. Pitts and W. McCulloch.
- Early successes (1956-1974) Perceptron by F. Rosenblatt
- First winter (1974-1980) Early optimistic promises could not be kept, problems are more difficult than expected
- Boom (1980-1987) New network concepts (e.g. Hopfield nets, Boltzmann machines) and a training method (backpropagation) evolve successfully, mostly driven by J. Hopfield, G. Hinton and D. Rumelhart
- Second winter (1990's) Classic economic bubble, severe damaged of reputation
- Uprising successes (2005-2017) Tedious application of engineering skills and tremendous increase in speed and capacity of computers lead to emerging success, thanks to Moor's law
- Big boom (2017-. . .) Transforms help to boost large language models to unprecedented heights . . .

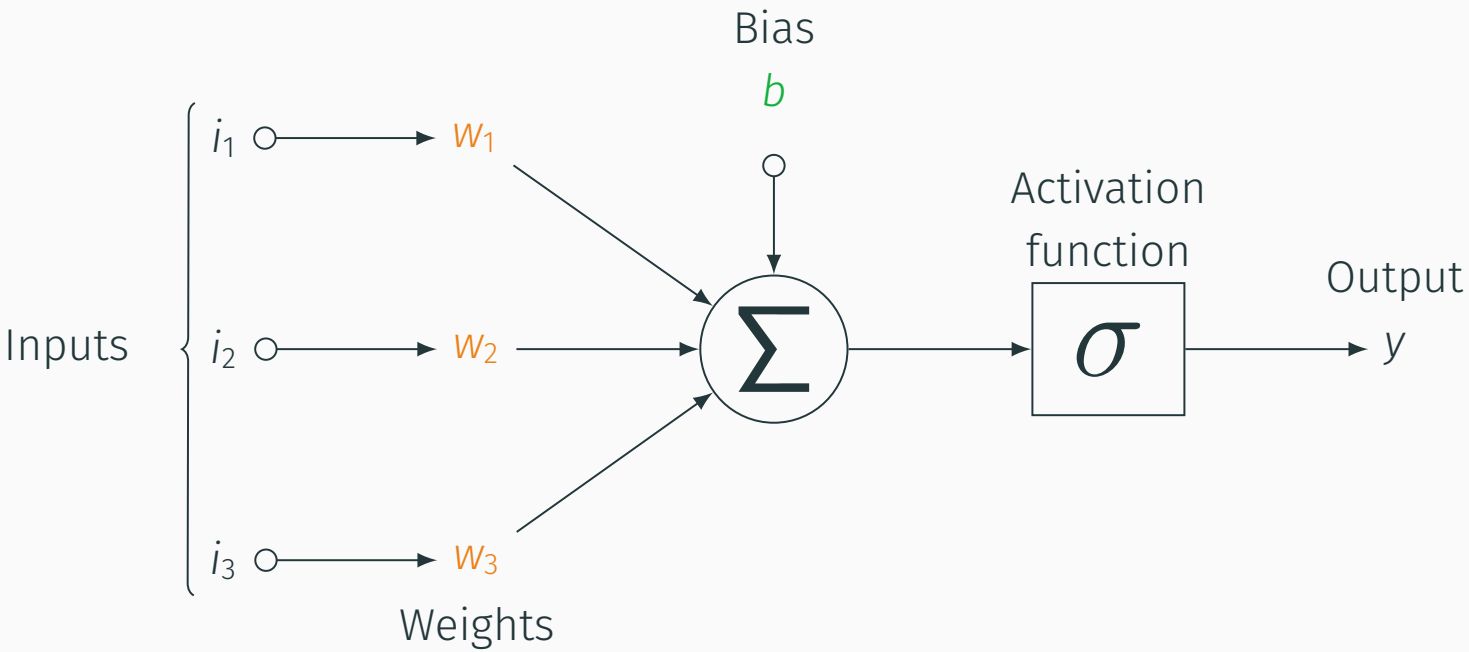
Network

- Layered concept, each layer consisting of a number of nodes
- Forward propagation of information
- Typical structure:



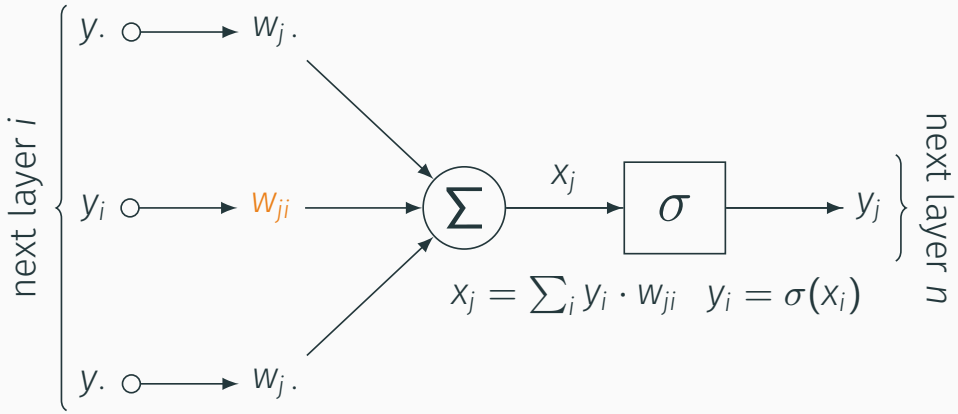
Nodes

- Nodes imitating neurons in biology
- Adapt weights w and biases b during 'training' by some kind of gradient descending method
- Typical structure:



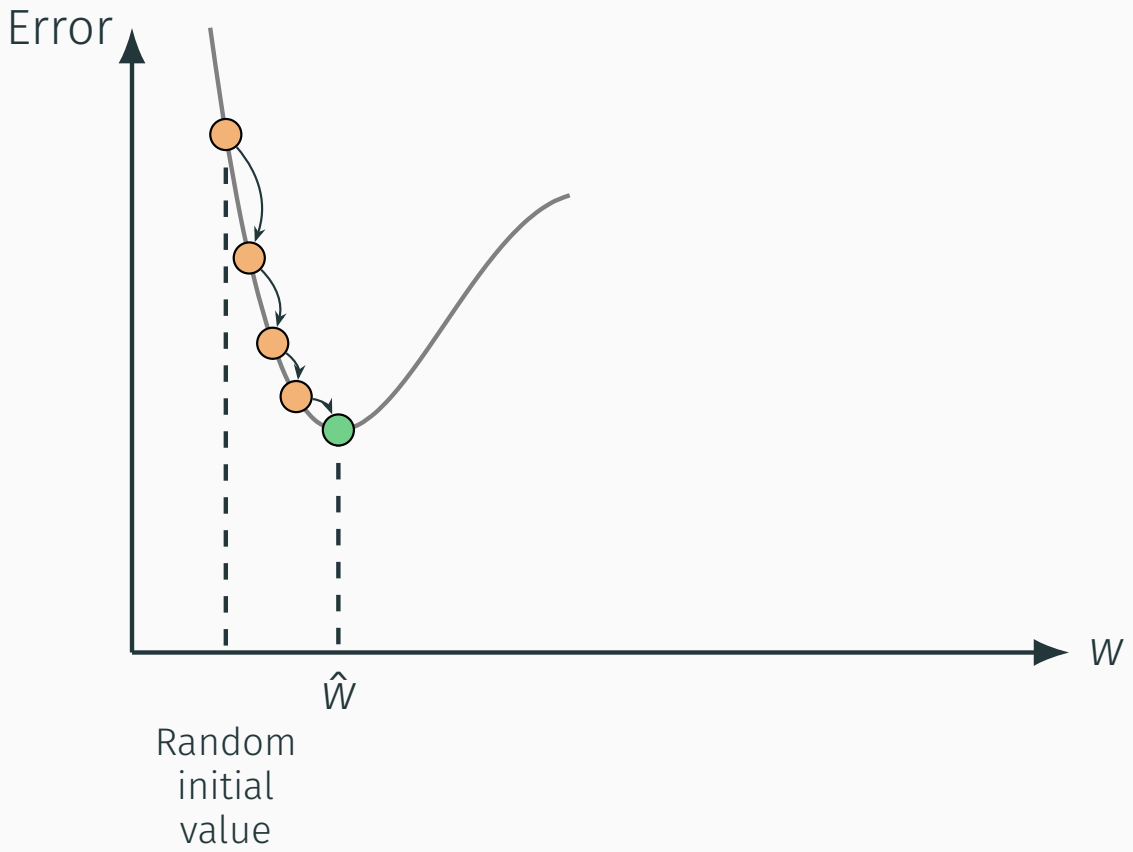
Training – how to adapt weights?

- Rumelhart, Hinton, Williams – Learning by back-propagating errors
- How is the error E influenced by the weight w_{ji} ? $\Rightarrow \frac{\partial E}{\partial w_{ji}}$
- With chain rule : $\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}}$
- $\frac{\partial x_j}{\partial w_{ji}} = y_i$
- $\frac{\partial y_j}{\partial x_j} = \frac{\partial \sigma(x)}{\partial x_j}$ e.g. for $\sigma(x) = \frac{1}{1+e^{-x}} \Rightarrow \frac{\partial \sigma}{\partial x_j} = y_j(1-y_j)$
- $\frac{\partial E}{\partial y_j} = \sum_n \frac{\partial E}{\partial x_n} \cdot w_{nj}$ of next layer n
- Propagate *backwards* from output with $E = \frac{1}{2}\sum(y-d)^2$ for target d
- Adapt weights : $\Delta w = -\eta \frac{\partial E}{\partial w}$



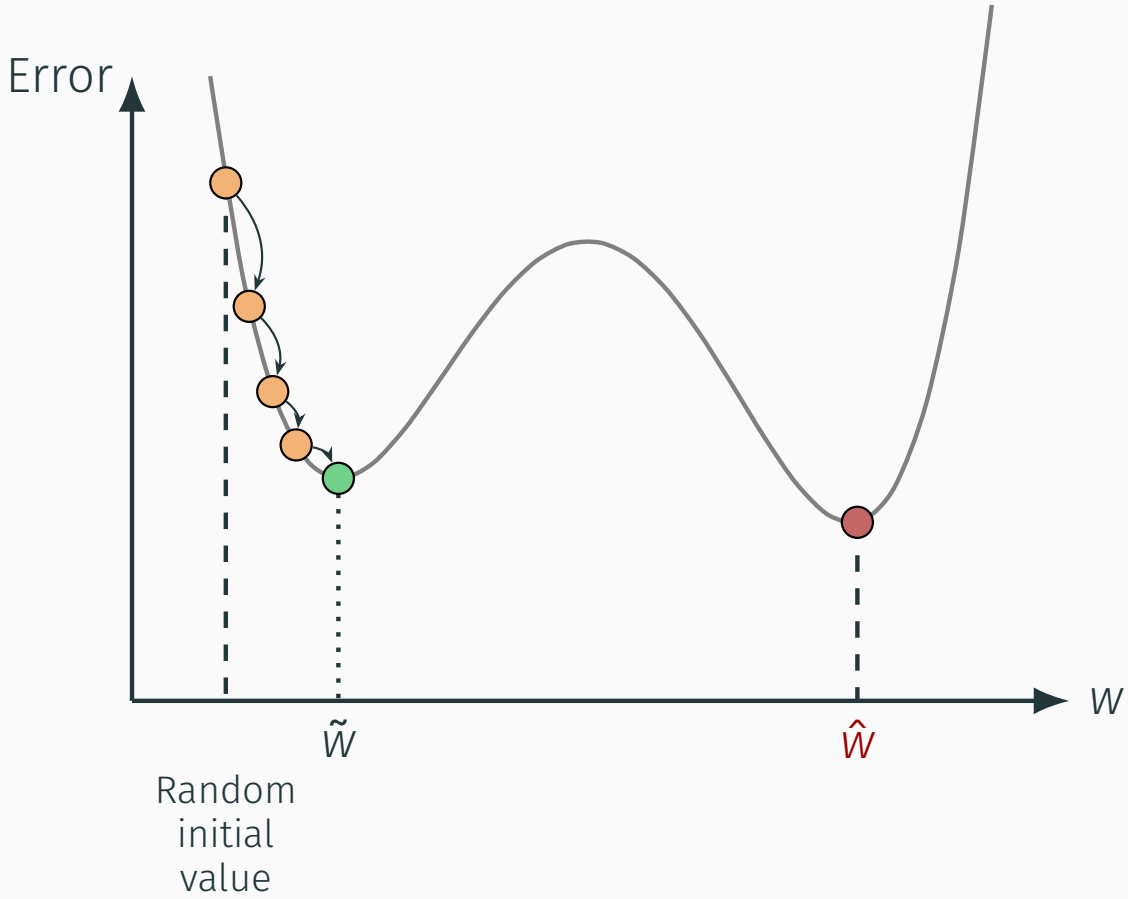
Gradient descent?

- Adapt weights : $\Delta w = -\eta \frac{\partial E}{\partial w}$
- Does this lead to an 'optimal' solution $\hat{w}$?



Global vs. local optima

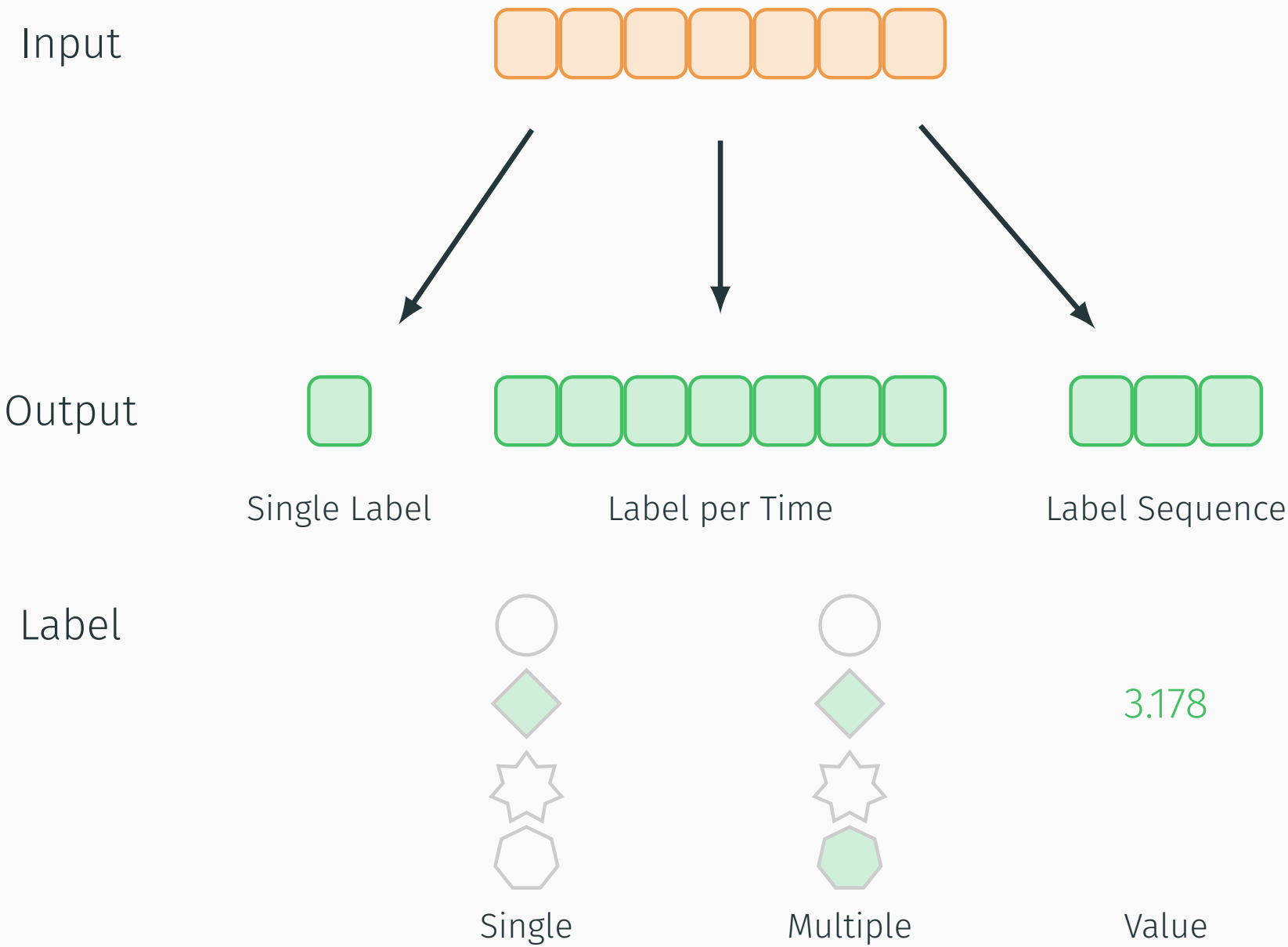
- Adapt weights : $\Delta w = -\eta \frac{\partial E}{\partial w}$
- Does this lead to an 'optimal' solution $\hat{w}$?



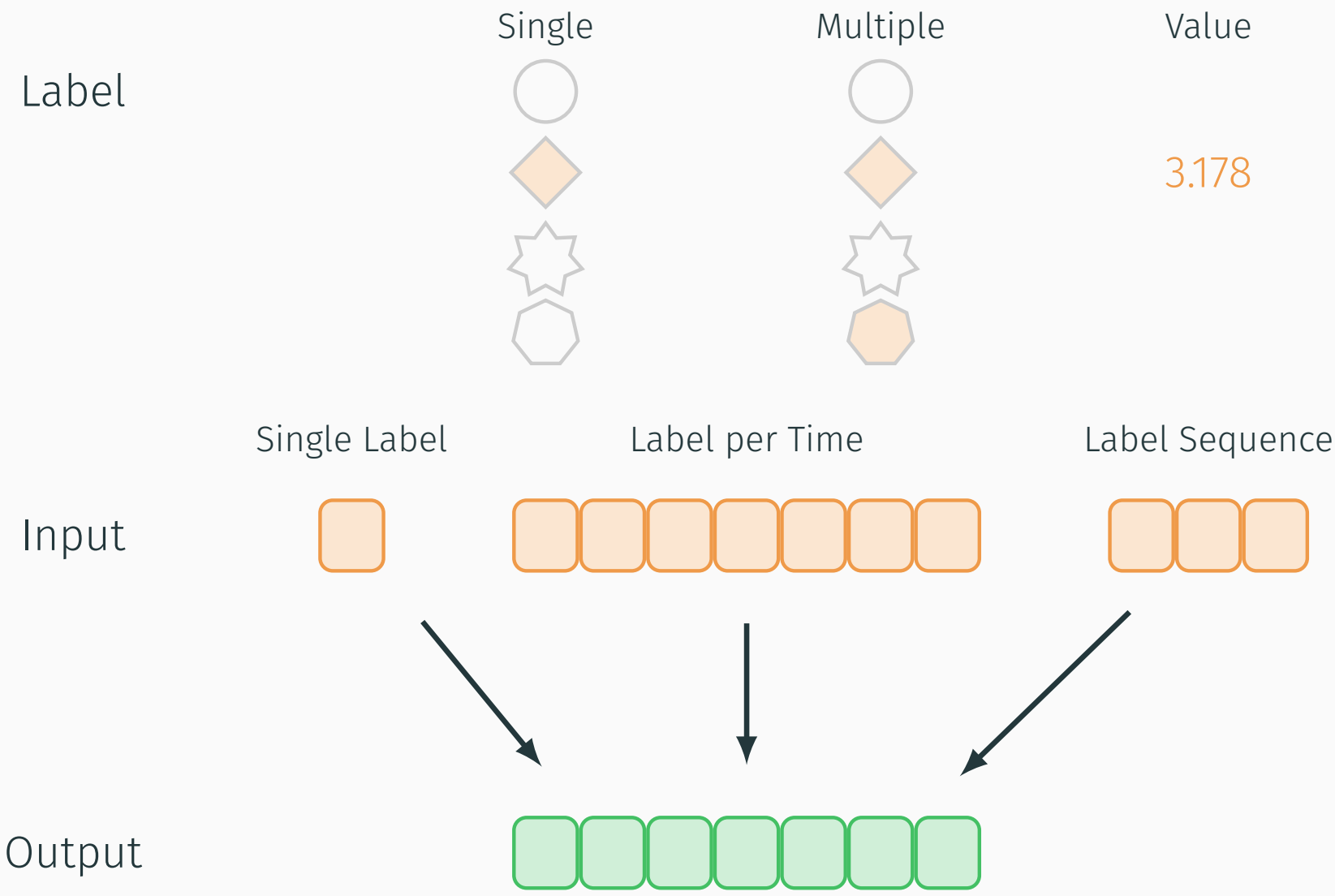
Although the concepts of neural networks are inspired by biological systems, they are not a replica of the mechanisms involved. This is particularly true for the learning process, which is certainly different from that used by organisms.

Schemes of Application

Analysis


Input
Output
Single Label
Label per Time
Label Sequence
Label
Single
Multiple
3.178
Value

Synthesis


Label
Single
Multiple
Value
3.178
Single Label
Label per Time
Label Sequence
Input
Output

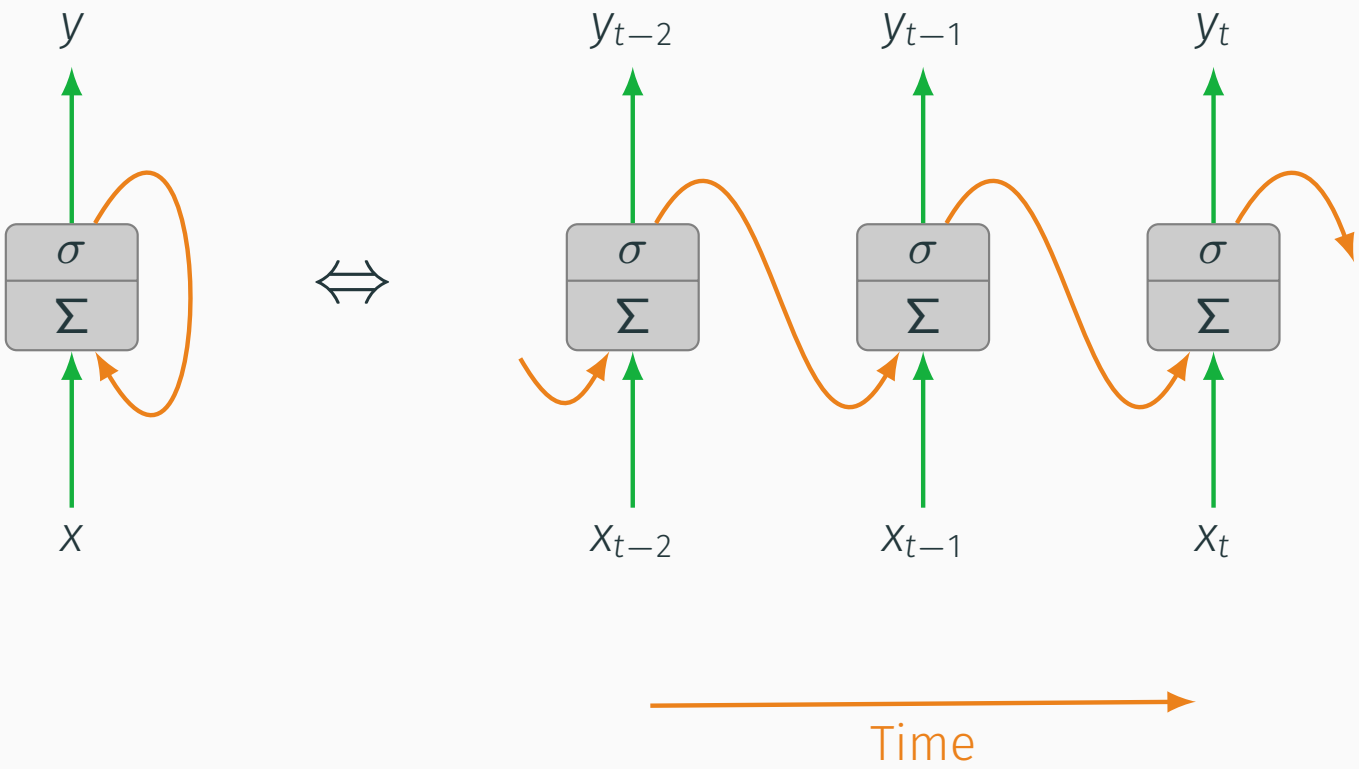
RNN (Recurrent Neural Network)

Motivation

- Traditional NN (Neural Network) have a 'feedforward' structure and therefore no memory of state.
- For the analysis of time series, i.e. data with a sequential internal structure, feedback proves to be very helpful and essential $\rightsquigarrow$ RNN (Recurrent Neural Network)

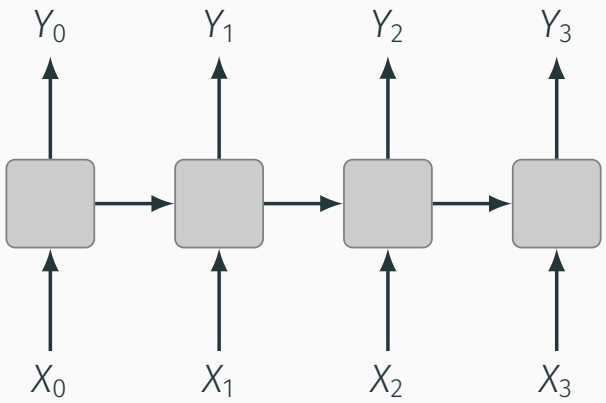
Recursive structure

Unrolling through time

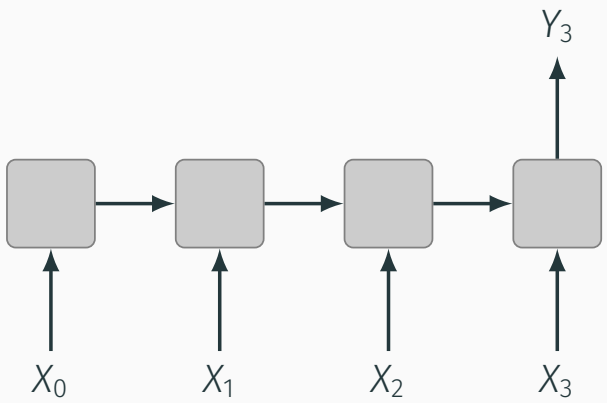


Applications

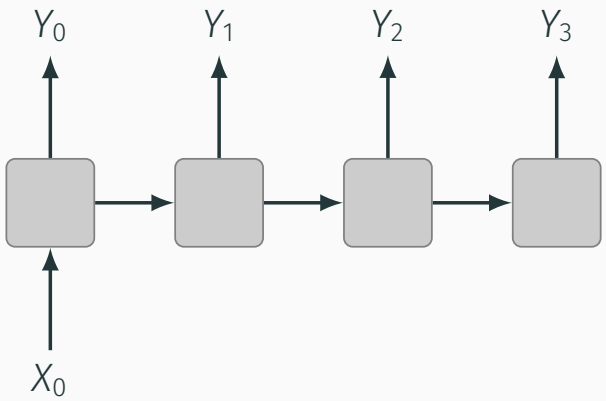
Sequence → Sequence



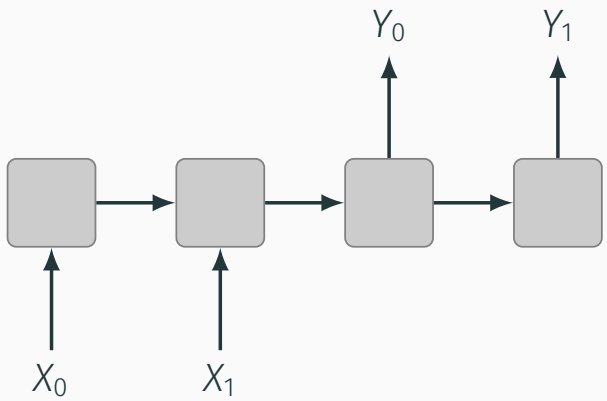
Sequence → Vector



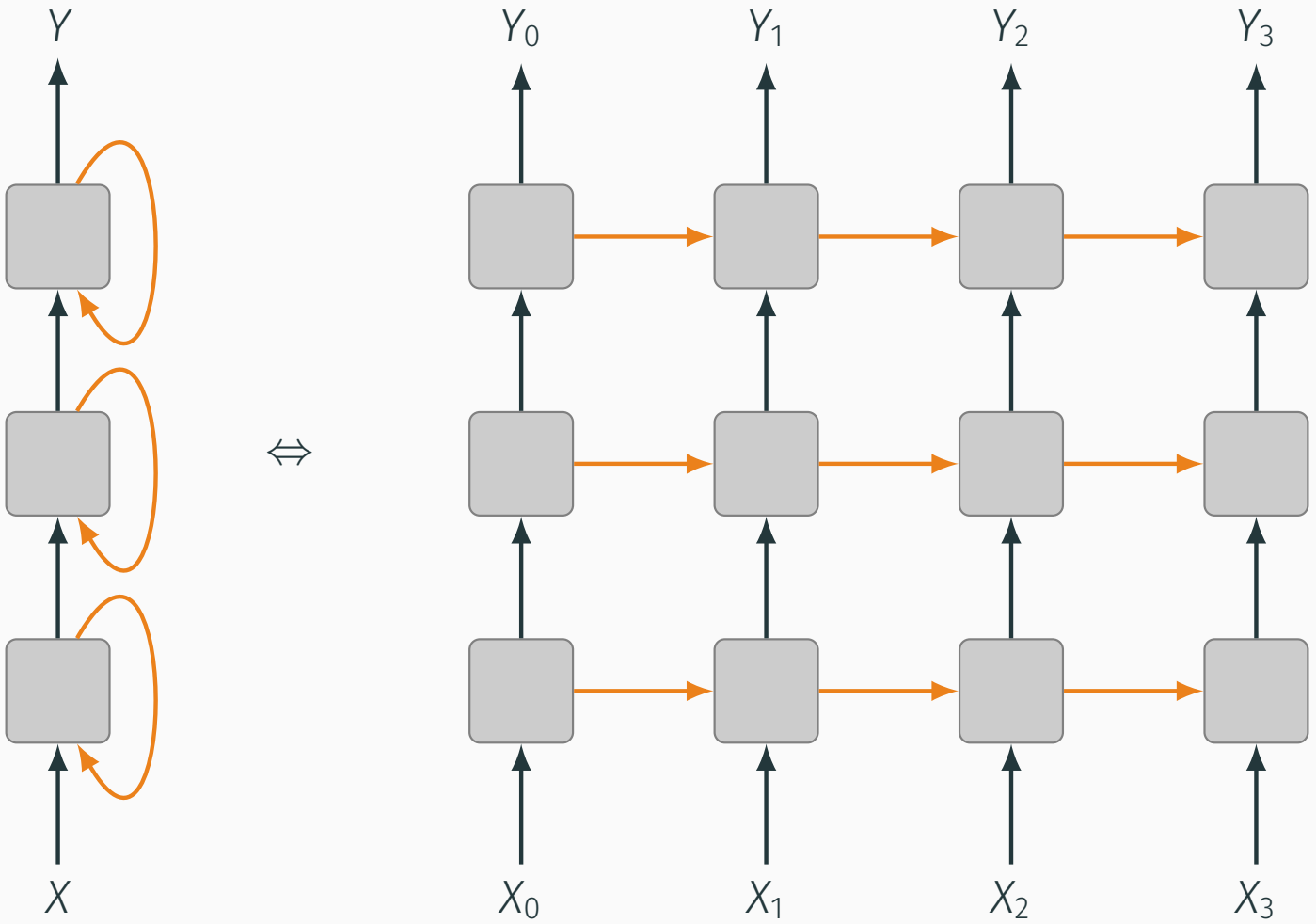
Vector → Sequence



Sequence → Sequence (delayed)



'Deep RNN'



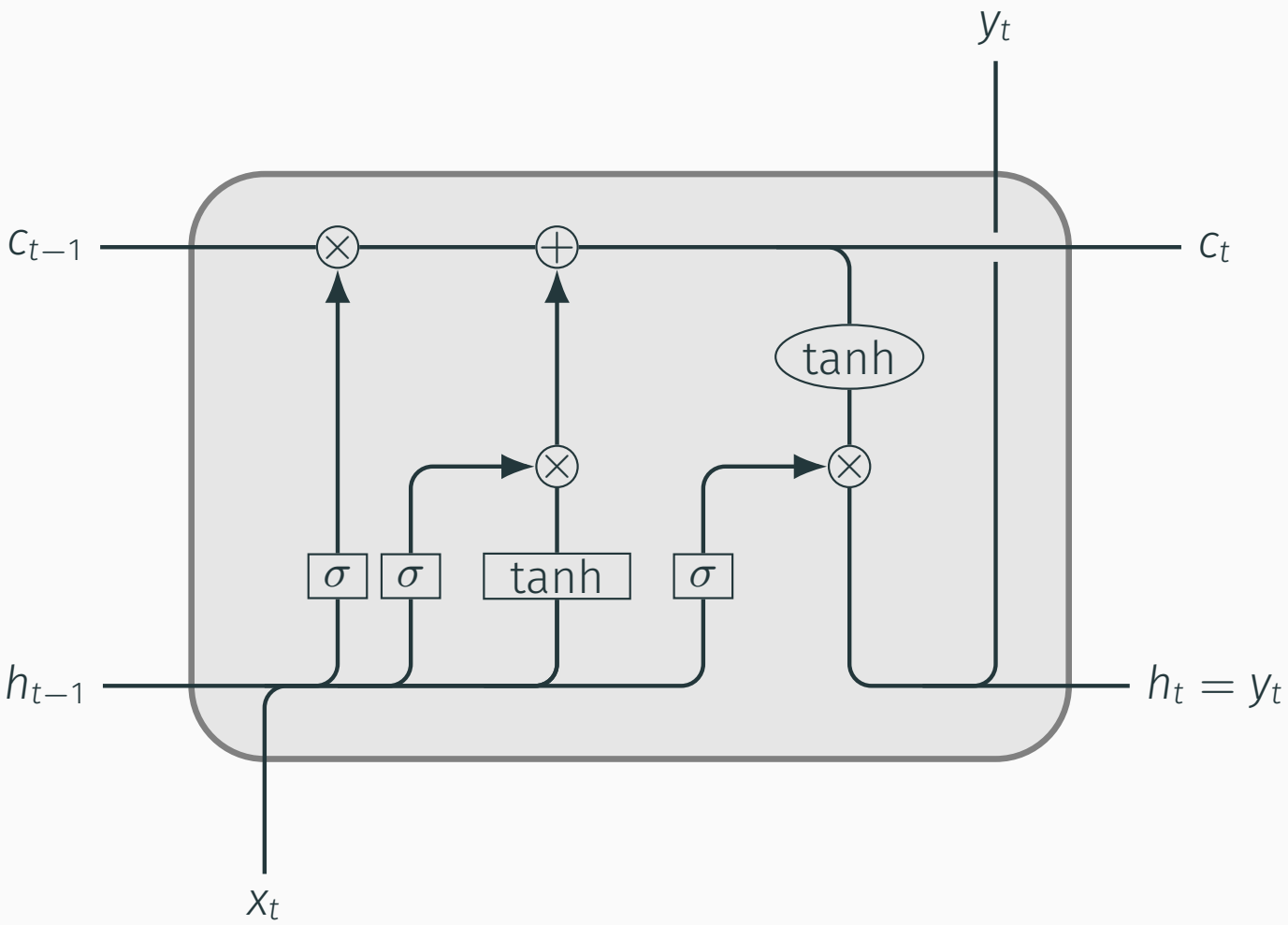
Training

- Training takes place by 'unrolling' through time and backward propagation of the error, BTT (Back-Propagation Through Time)
- Problems: disappearing / exploding / oscillating gradients
- Solution concepts:
  - non-saturating activation function
  - 'good' initialization
  - batch normalization
  - gradient limitation
  - time step limitation
  - ...

Modell - LSTM (Long Short Term Memory)

- Foundation: Sepp Hochreiter, Jürgen Schmidhuber 1997

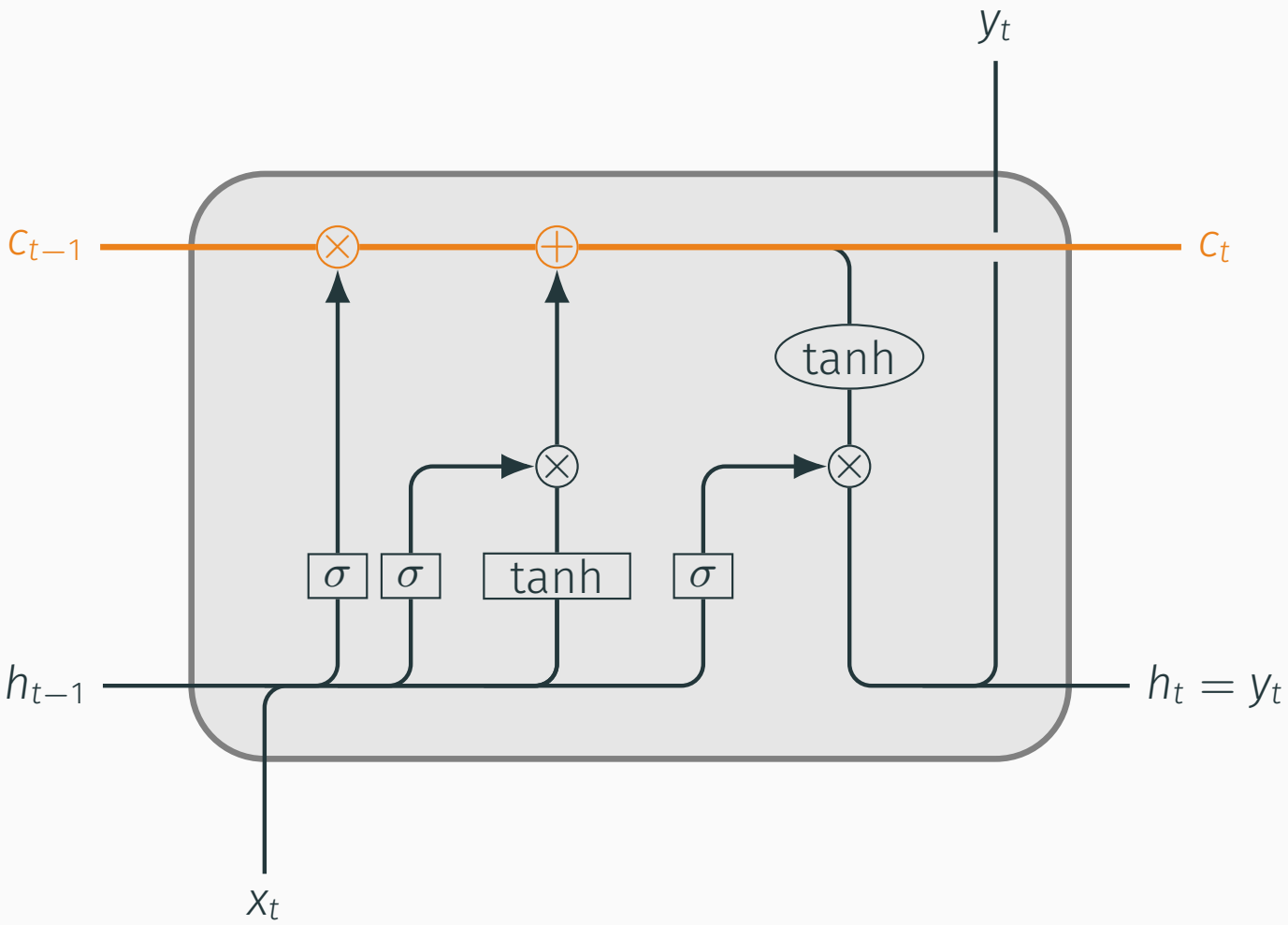
  . . . subsequently adapted, optimized and expanded many times



LSTM – I

- 'Long-term State' c

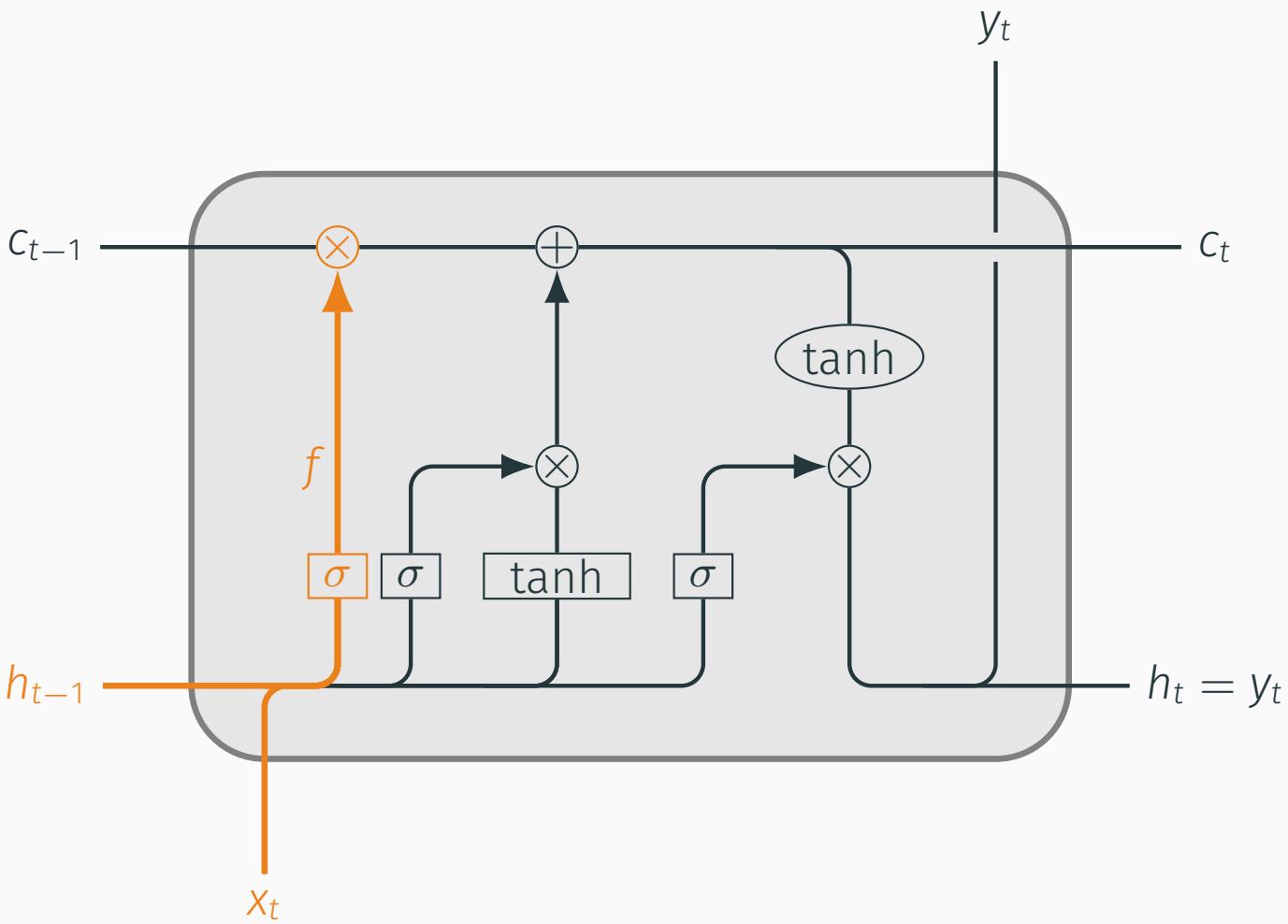
  Operations : $\otimes \rightsquigarrow$ forget, $\oplus \rightsquigarrow$ add



LSTM – II

- 'Forget' f : what shall be deleted from memory c_{t-1}?

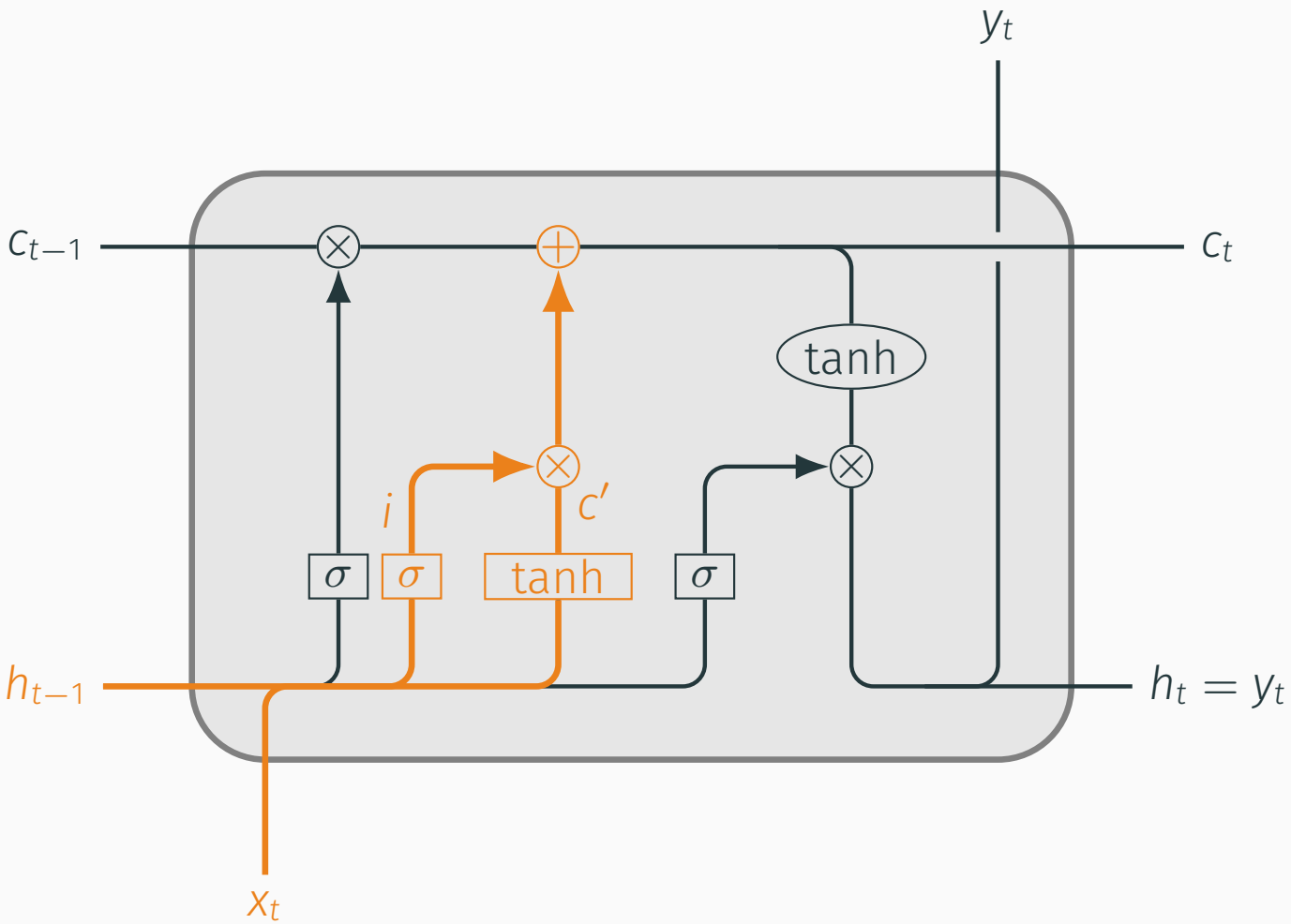
  $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$



LSTM – III

- 'Input' i : what shall be added to memory c_t?

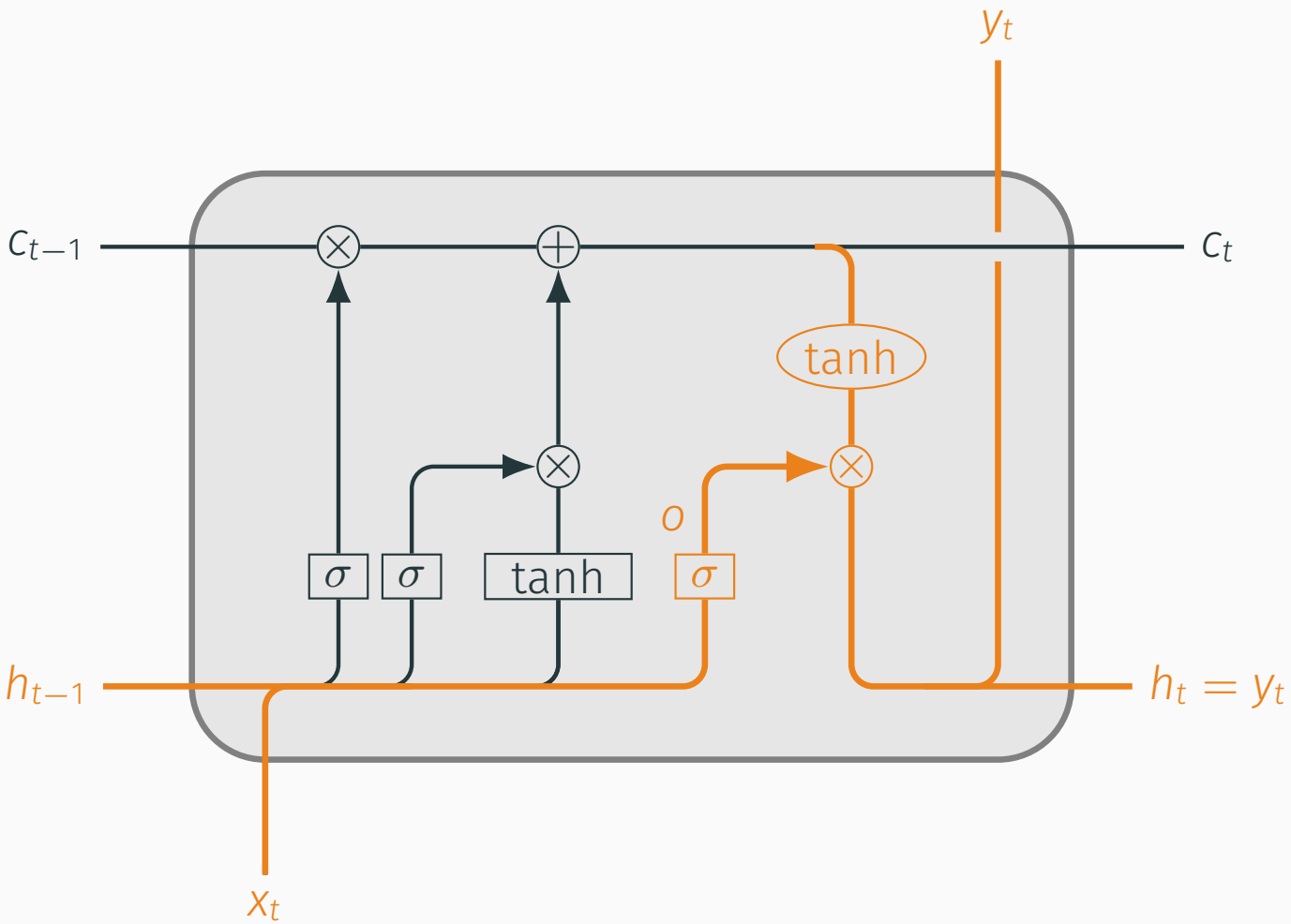
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad \text{and} \qquad c'_t = \mathbf{tanh}(W_c \cdot [h_{t-1}, x_t] + b_c)$$



LSTM – IV

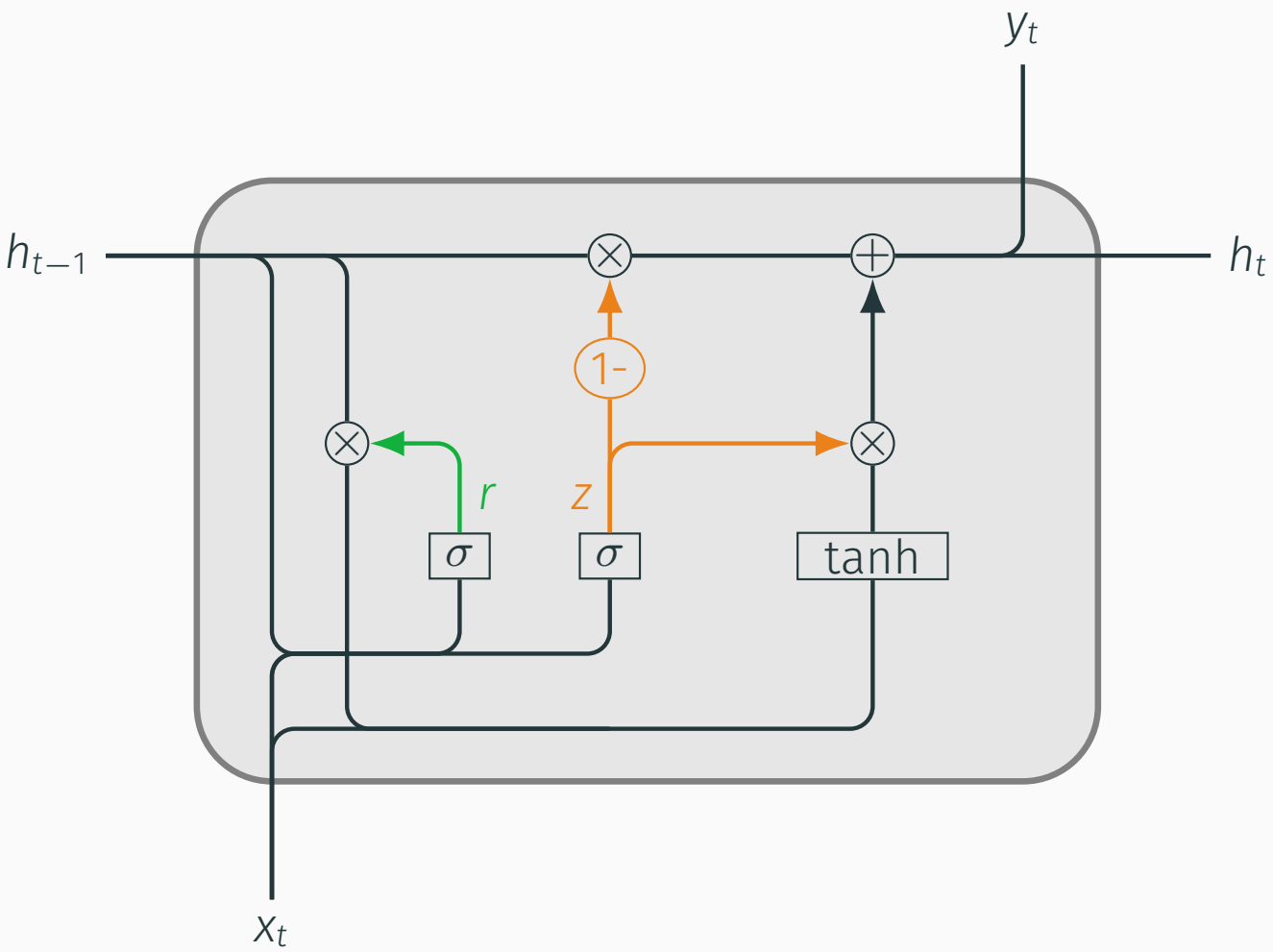
- 'Output' o : what shall be used as output y_t?

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \qquad \text{und} \qquad y_t = o_t \otimes \tanh(c_t)$$



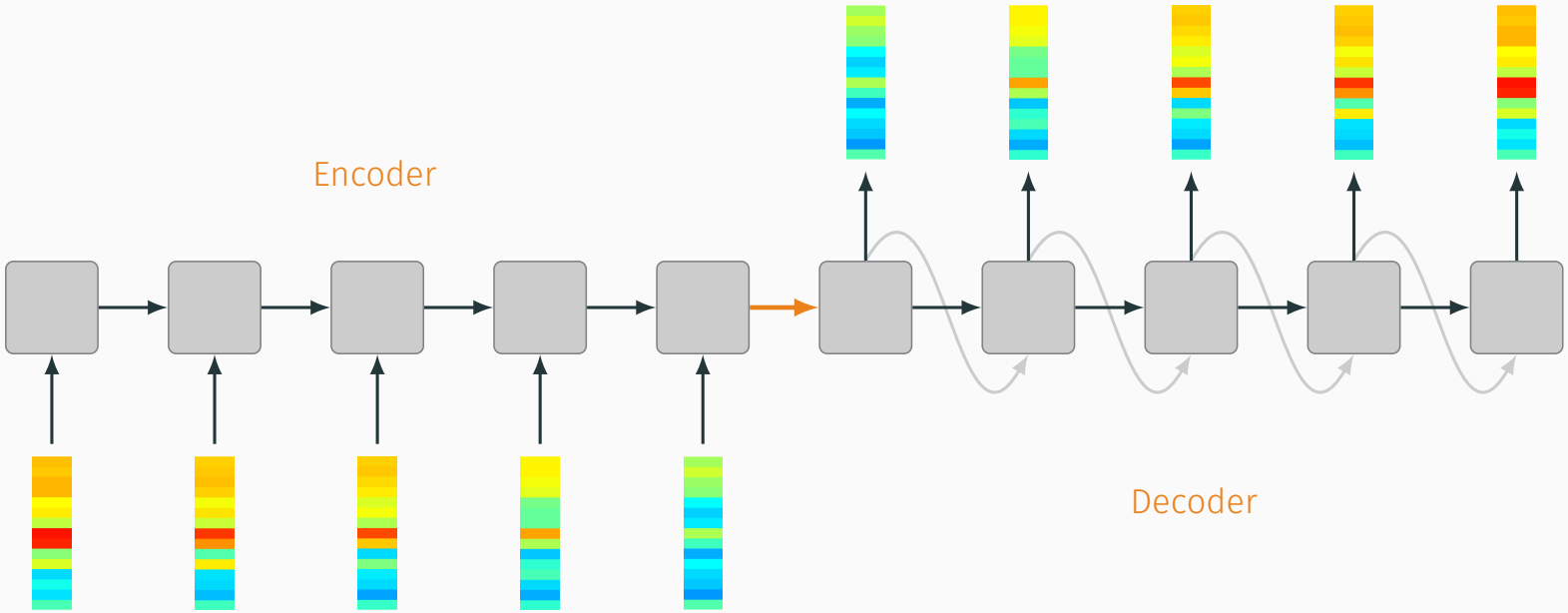
Modell - GRU (Gated Recurrent Unit)

- Modifications : 'update' z and 'reset' r, elimination of c'
- Performance comparable to LSTM, but simpler structure



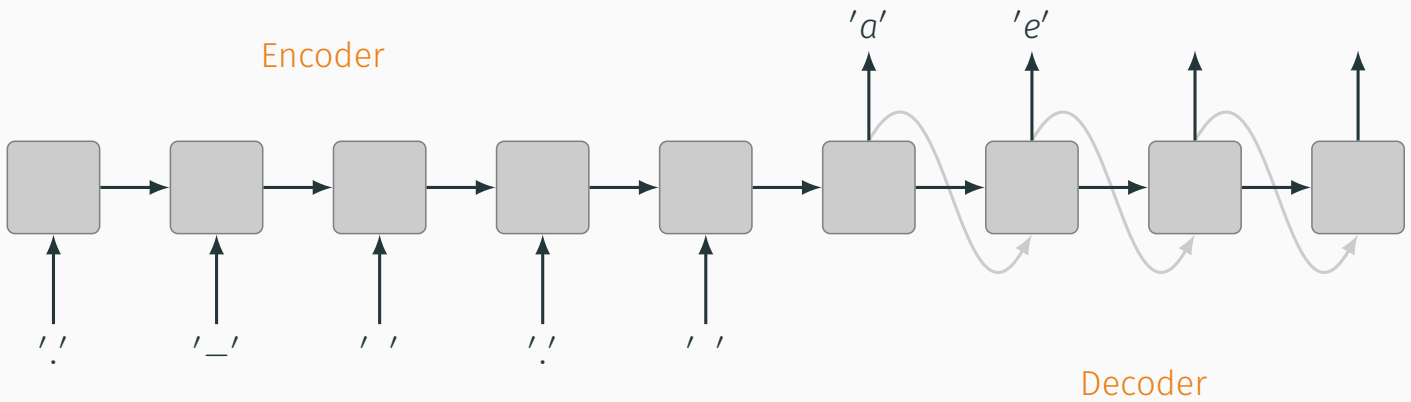
Example I – Diagnostics

- Discriminate normal behaviour from malfunctions
- Application to ’noise’ of elevators
- Spectrum over time as input data, re-construction backwards as output
- Autoencoder : Encoder $\rightarrow$ Decoder enforces compression to *one* state



Example II – Morse code

- Translate Morse code to plain text
- Modell 'Sequence to Sequence Learning'
- Fixed length sequences for the sake of simplicity
- Coding of ASCII symbols required ('Embedding')



Code Snippets

LSTM

- Basing in Tensorflow with Keras frontend

```
$ pip install tensorflow keras
```

- Create model :

```
latent_dim = 200
model = Sequential()
model.add(layers.LSTM(latent_dim, input_shape=(max_len_x,len(chars_in))))
model.add(layers.RepeatVector(max_len_y))
model.add(layers.LSTM(latent_dim, return_sequences=True))
model.add(layers.TimeDistributed(layers.Dense(len(chars_out))))
model.add(layers.Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

- Train model :

```
hist = model.fit(x_train, y_train, batch_size=Batch_size, epochs=Epochs, validation_data=(x_val, y_val
```

- Test model :

```
pre = model.predict(x)
```

Addendum

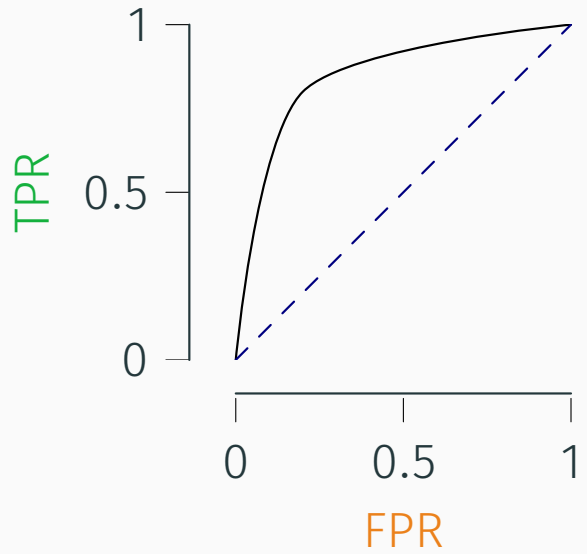
Characterizing Binary Classifier

ROC – Receiver Operating Characteristic

Plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

true positive	false negative
false positive	true negative

Typical classifier vs random choice



References

- Turing – Computing Machinery and Intelligence
- Rumelhart, Hinton, Williams – Learning by back-propagating errors
- Hochreiter, Schmidhuber – LSTM
- Comparison of different RNN
- Sequence to Sequence Learning
- Backpropagation and the Brain
- Generative Pre-Training

Exercise

Your task

Back to the roots! Implement a simple Morse decoder without using high-level libraries. You can limit the decoder to a few characters, e.g. A : ·−, B : −· · ·, C : −·−·, D : −··, E : −·.

1. How can the input sequence be represented at the input nodes? How to encode the symbols, i.e. dot, dash and void? Describe a possible solution.
2. Implement a simple network with one intermediate layer of a few neurons in ‘plain’ Python. The output can be represented by a ‘one hot encoding’.
3. Work out the details of the back-propagtion of the error according to the activation function choosen. Implement the adaption by back-propagation in Python. How can the error be determined?
4. Implement a training & testing environment for the characters choosen und train the network.
5. Summarize your experiences, findings and insights in short essay.
6. Post everything to the mailbox on Ilias.

Rating : one point for each task marked red and one for an extraordinary treatment (e.g. extension of functionality).