RNNs für Zeitreihenvorhersagen

Antonio Rosolia

ZHAW

November 27, 2020

Overview

Zeitreihenvorhersagen

- Informationen aus Zeitreihen extrahieren und nächsten Wert abschätzen
- Typischerweise Periodizität und Trends
- Überall anzutreffen:
  - Business: Web Traffic, Supply Chain, Buchungssysteme, ...
  - Finanzwesen: Econometrics, Exchange, Aktien, ...
  - Wissenschaft: Wettervorhersage, Erdbebenerkennung, ...
  - Engineering: Sensorik, Predictive Maintenance, ...
  - Medizin: Diagnosen, Monitoring, ...

Modelle

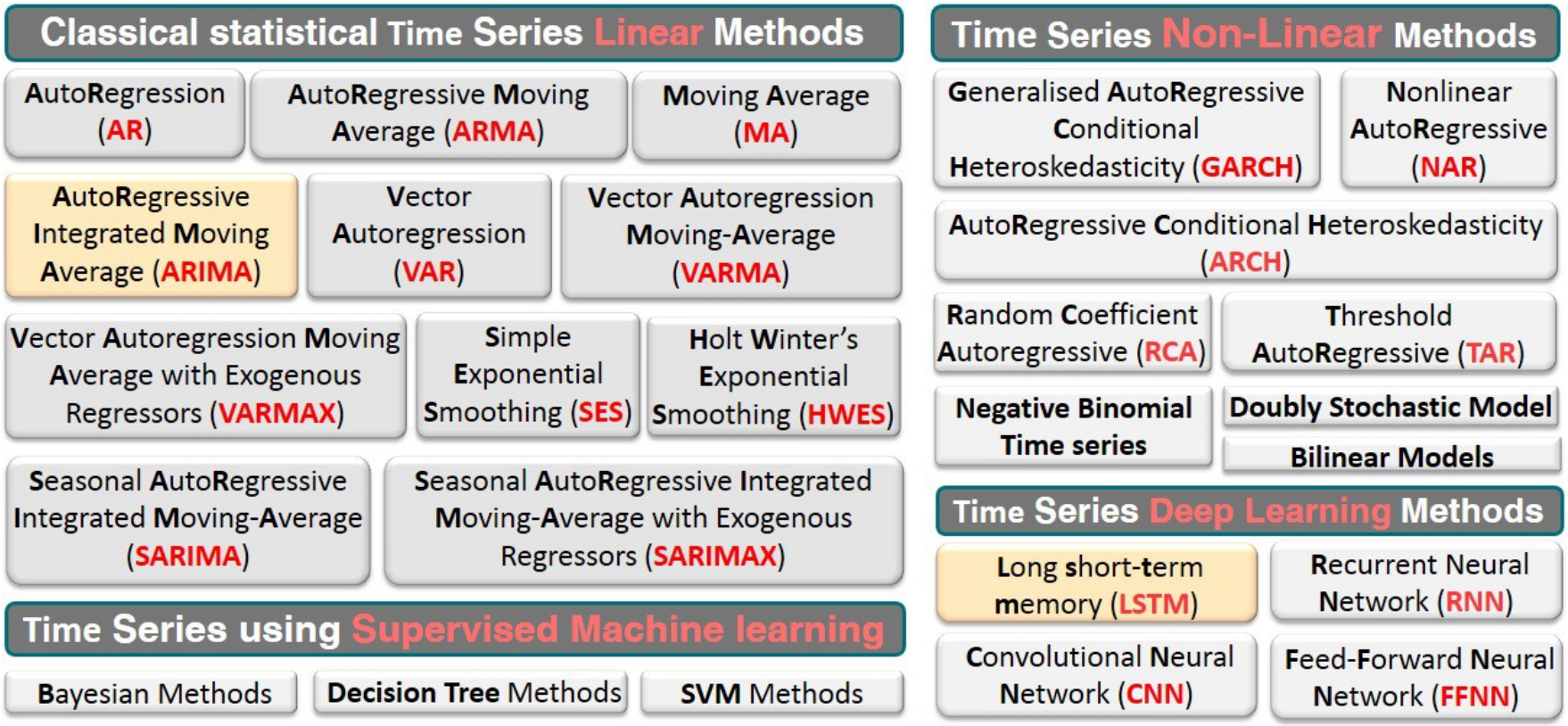
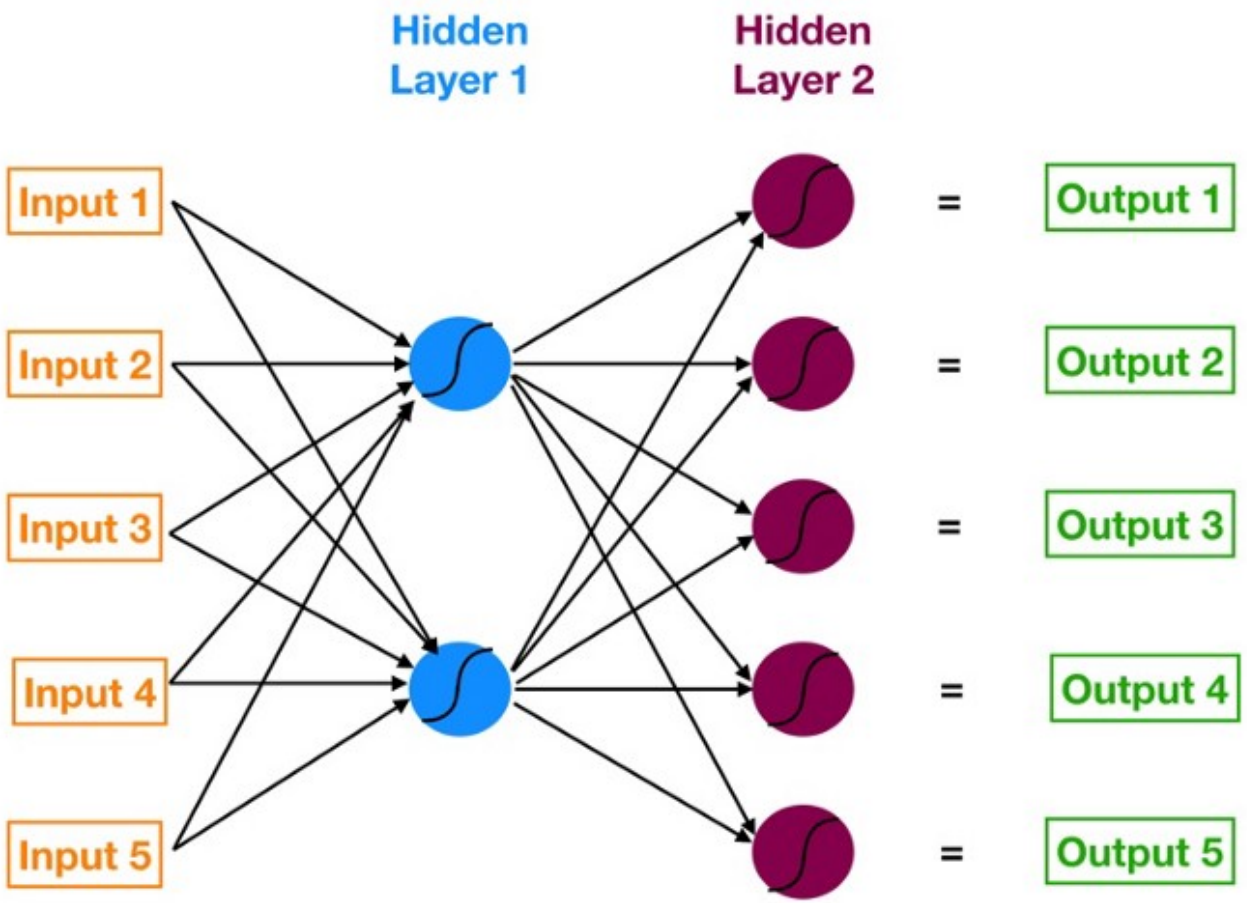


Figure: Überblick über Bekannte Modelle

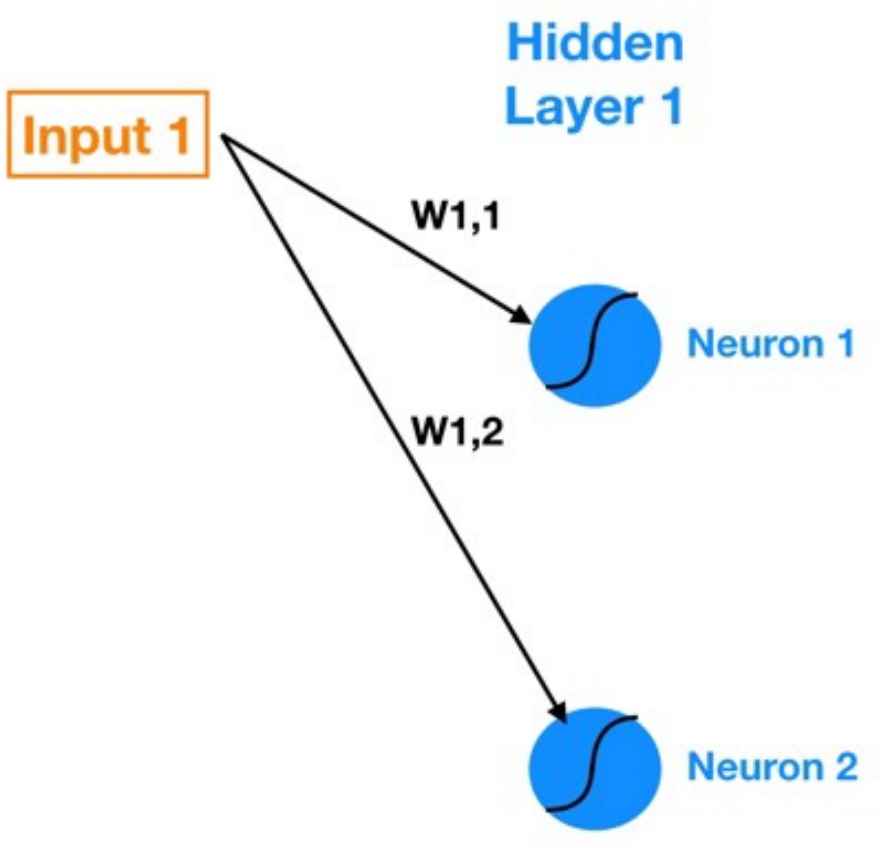
Bild von: https://medium.com/@fenjiro/time-series-for-business-a-general-introduction-50968346e660 [aufgerufen 30 Oktober 2020].

Einschub Neuronale Netzwerke - Überblick



Nachfolgende Bilder von: https://towardsdatascience.com/understanding-neural-networks-19020b758230 [aufgerufen 04 November 2020].

Einschub Neuronale Netzwerke - Input Layer



Einschub Neuronale Netzwerke - Forward Propagation

$$Z1 = W1 * In1 + W2 * In2 + W3 * In3 + W4 * In4 + W5 * In5 + BiasNeuron1$$

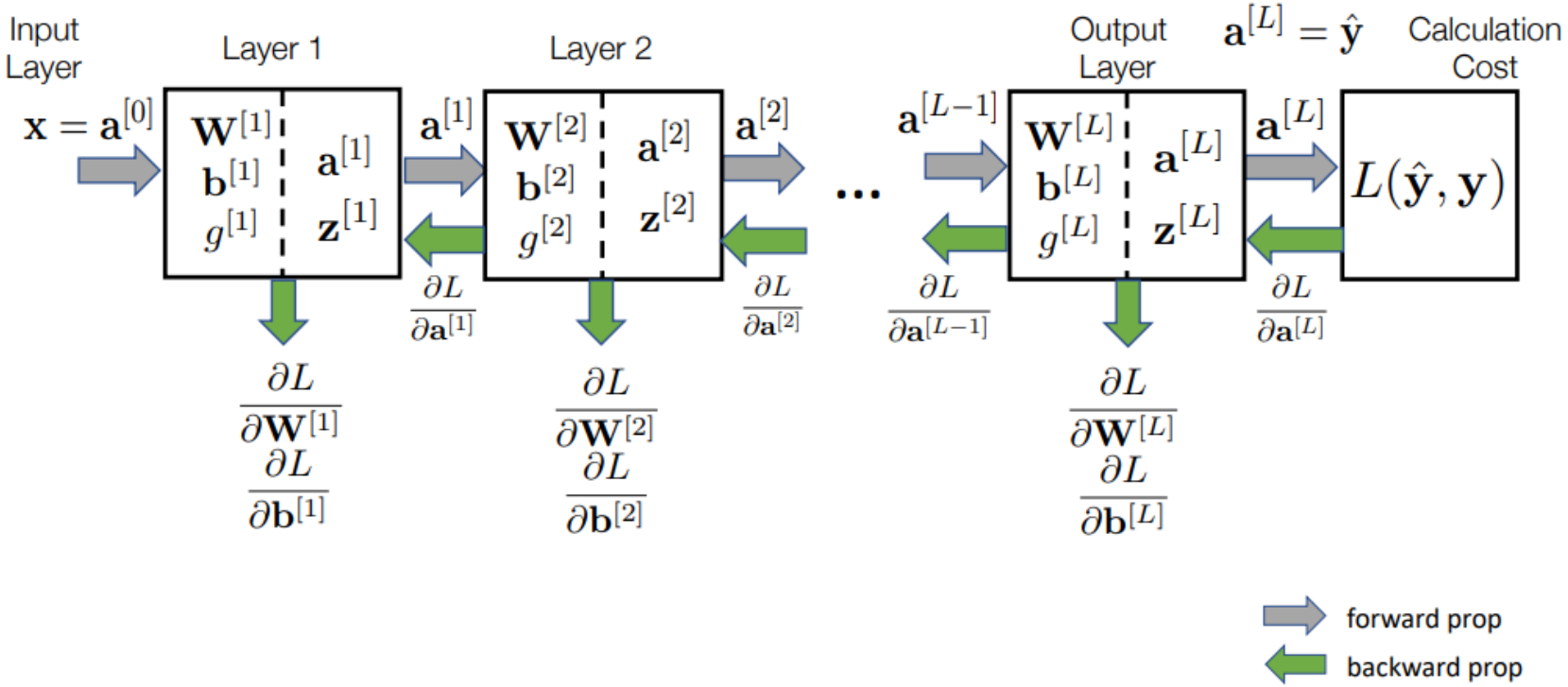
$$\begin{bmatrix} W1,1 & W2,1 & W3,1 & W4,1 & W5,1 \\ W1,2 & W2,2 & W3,2 & W4,2 & W5,2 \end{bmatrix} \times \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \end{bmatrix} + \begin{bmatrix} Bias1 \\ Bias2 \end{bmatrix} = \begin{bmatrix} Z1 \\ Z2 \end{bmatrix}$$

$$Neuron1Activation = Sigmoid(Z1)$$

Einschub Neuronale Netzwerke - Kostenfunktion

- MSE $MSE = Sum[(Prediction - Actual)^2] * (1/numObservations)$
- $J(\theta) = \frac{1}{N} \sum_{i=0}^{N} (y_i - \hat{y}_i)^2$; $\theta = (W, b)$
- Minimierung der Kostenfunktion J mittels Gradient Descent und Backpropagation

Einschub Neuronale Netzwerke - Backward Propagation



$$\mathbf{W}^{[l]} \leftarrow \mathbf{W}^{[l]} - \alpha \frac{\partial J}{\partial \mathbf{W}^{[l]}}$$

$$\mathbf{b}^{[l]} \leftarrow \mathbf{b}^{[l]} - \alpha \frac{\partial J}{\partial \mathbf{b}^{[l]}}$$

Warum RNNs

- Artificial Neural Network kann sich nicht erinnern was es lernt
  - jede Iteration startet es neu
- RNNs haben Memory
  - Kontext aus Sequenzen
  - Ideal um patterns und Korrelationen zu finden

Viele Anwendungsbereiche

- Speech Recognition
- Sentiment Classification
- Machine Translation
- Captioning, subtitling

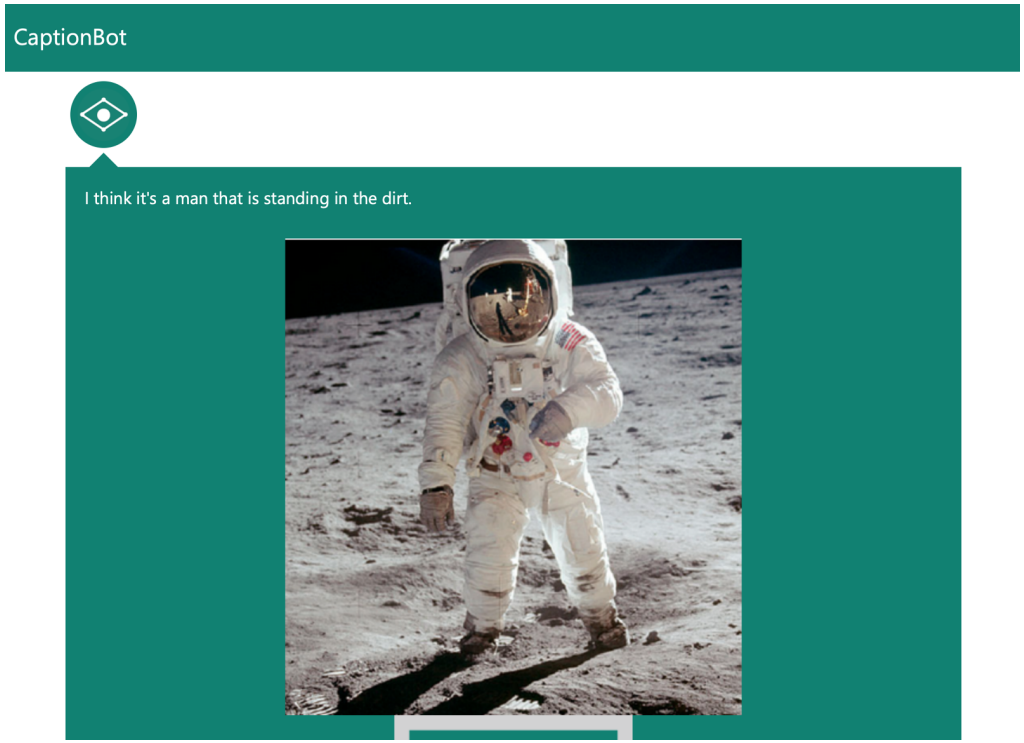
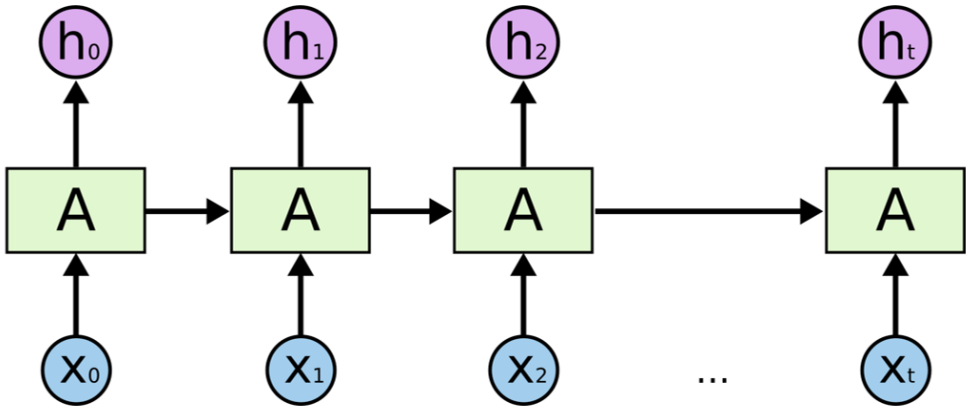


Figure: Captionbot @Microsoft

Viele Anwendungsbereiche

- Chatbots
  - Google Duplex (Blogeintrag zum Friseurtermin)
- Text und Musik Generator
  - DeepJazz
- Vorhersagen für Zeitreihen

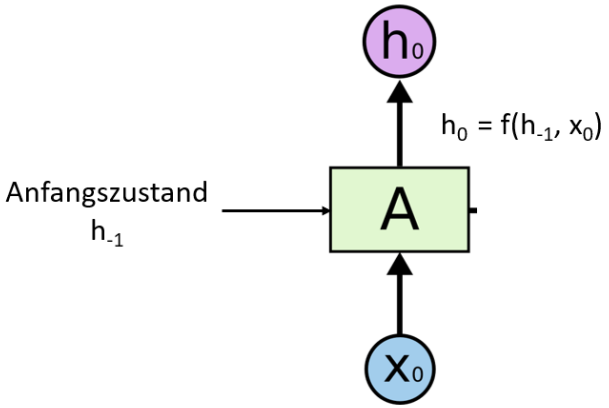
RNN Intuition - 1



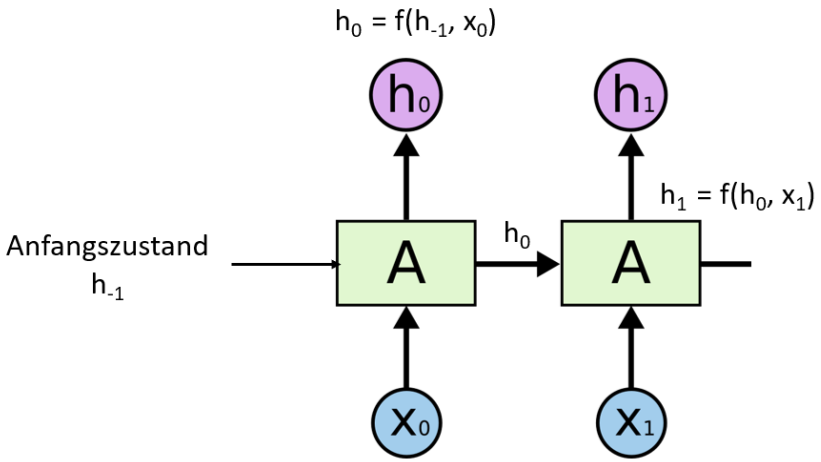
- Vorheriger Zustand wird wieder gebraucht, Informationen werden weitergegeben

Nachfolgende Bilder von: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [aufgerufen 4 Oktober 2020].

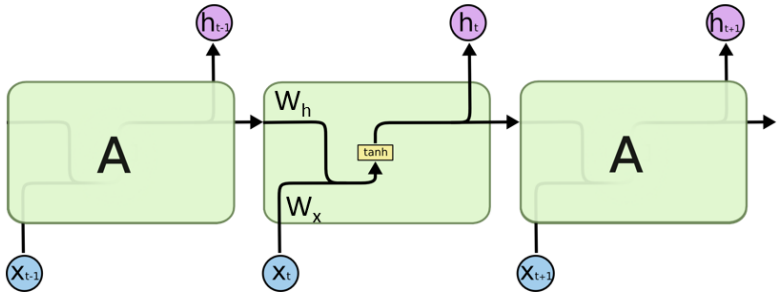
RNN Intuition - 2



RNN Intuition - 3



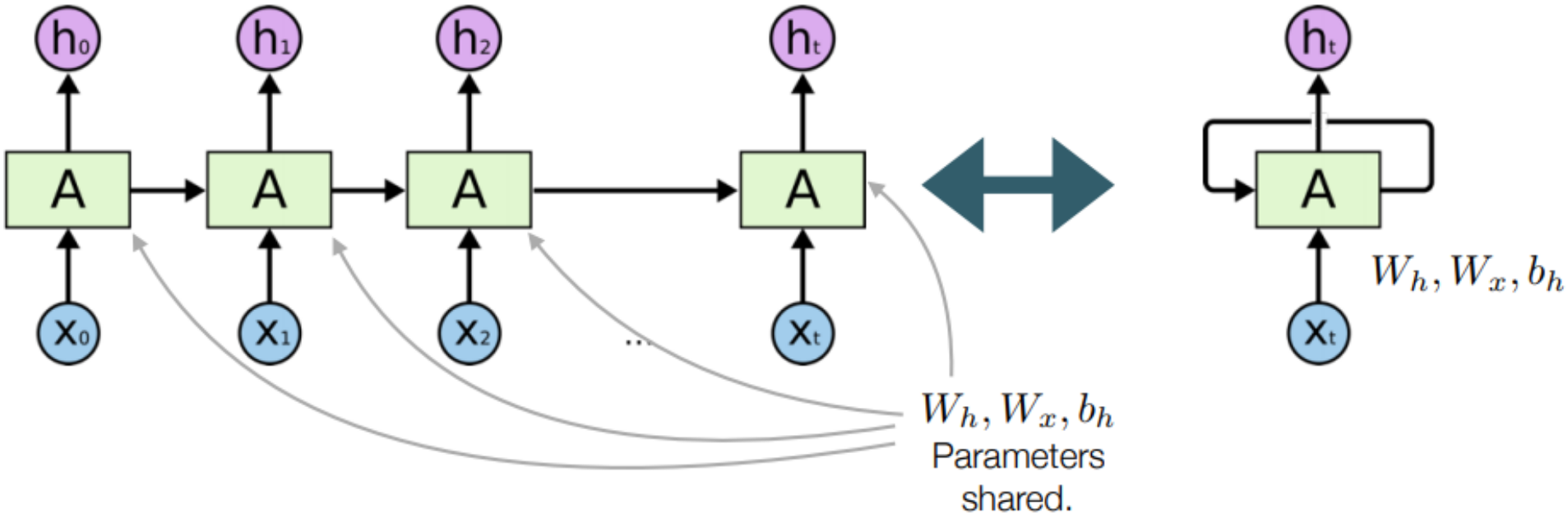
RNN Intuition - 4



$$h_t = g(W_x * x_t + W_h * h_{t-1} + b_h)$$

- W sind die Weight Matrix
  - W_h: wheight x wheight
  - W_x: wheight x Anzahl Features
  - b_h: wheight x 1
- g ist nichtlineare Aktivierungsfunktion (tanh)

RNN Intuition - 5



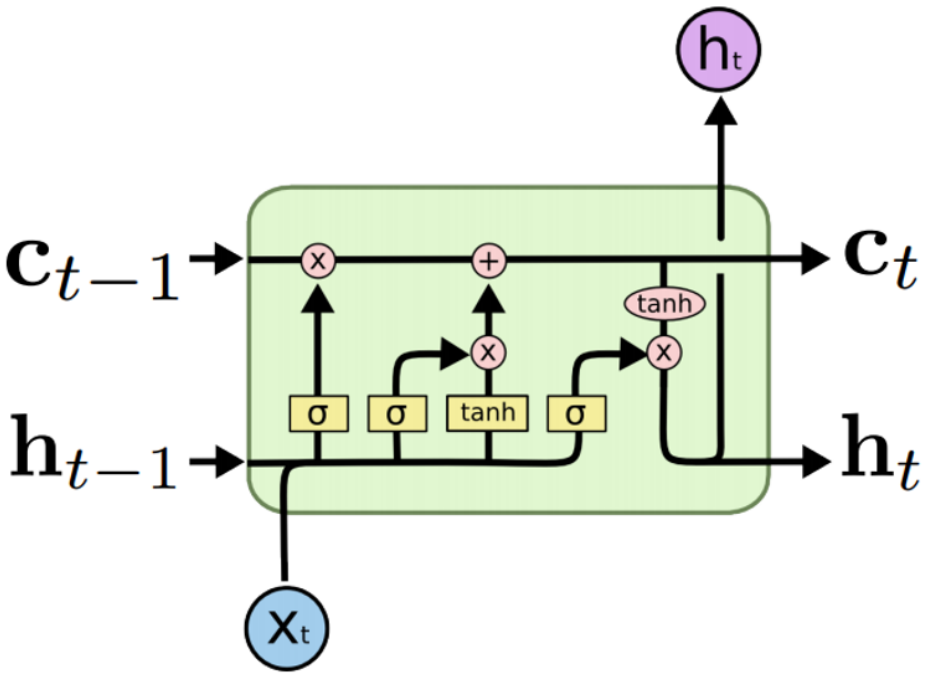
- Parameter W_x, W_h und b_h für alle Zellen gleich!

Simple RNN - Problem of Long-Term Dependencies

- Einfache RNNs funktionieren gut für kürzliche Vergangenheit
- Theoretisch möglich auch grösseren Kontext
- Praktisch problematisch bei grösserem Kontext
  - Problem entdeckt von Hochreiter (1991) und Bengio, et. al (1994)
  - Vanishing Gradient und Exploding Gradient
- Um Problem zu umgehen LSTM enworfen von Hochreiter und Schmidhuber (1997)

LSTM

- Sehr populär
- Spezifisch designt für Long-Term dependency Problem
- Gleiche Verkettung wie bei simplen RNNs
- Komplexere Architektur als simple RNNs

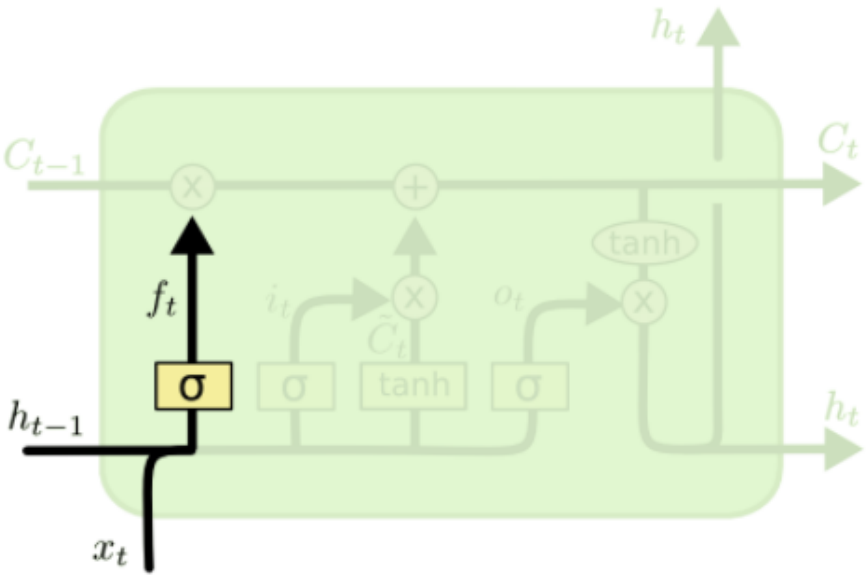


long-term memory

short-term memory
working memory

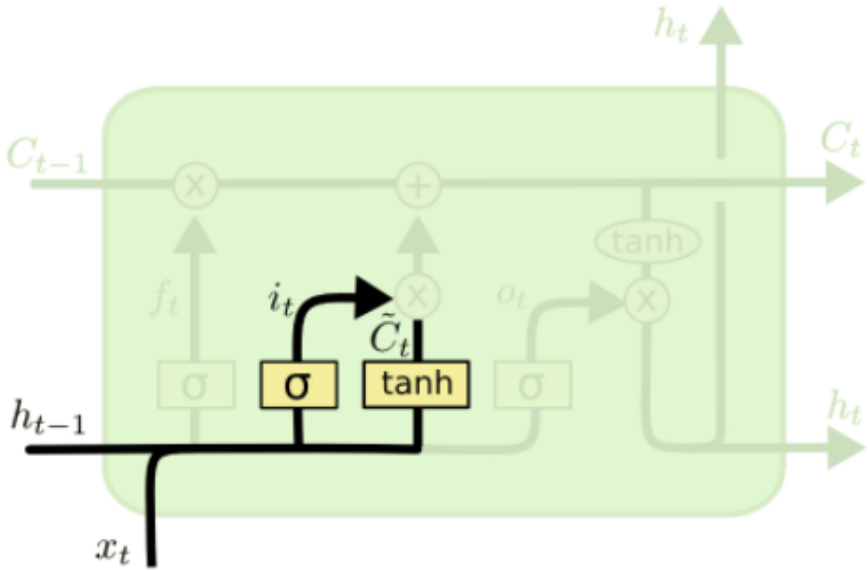
Nachfolgende Bilder von: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [aufgerufen 4 Oktober 2020].

LSTM Architektur - Forget Mechanism



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

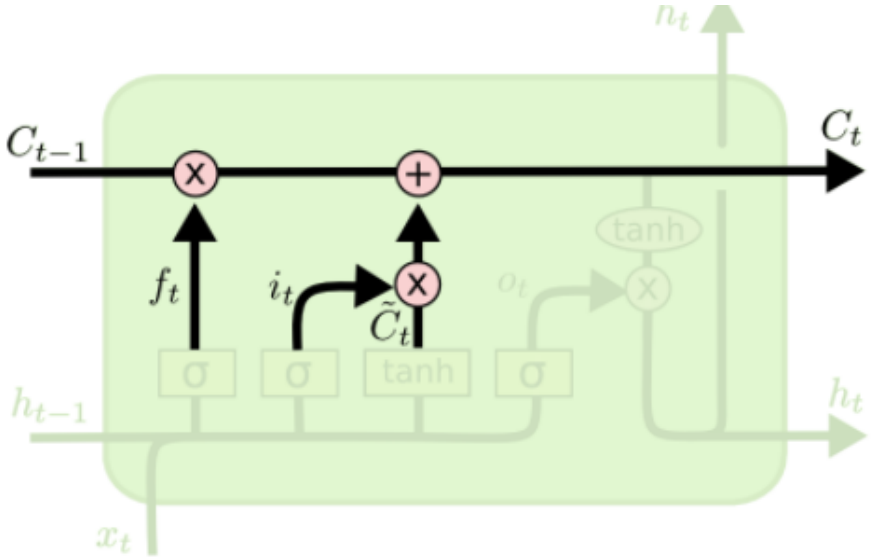
LSTM Architektur - Save Mechanism



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

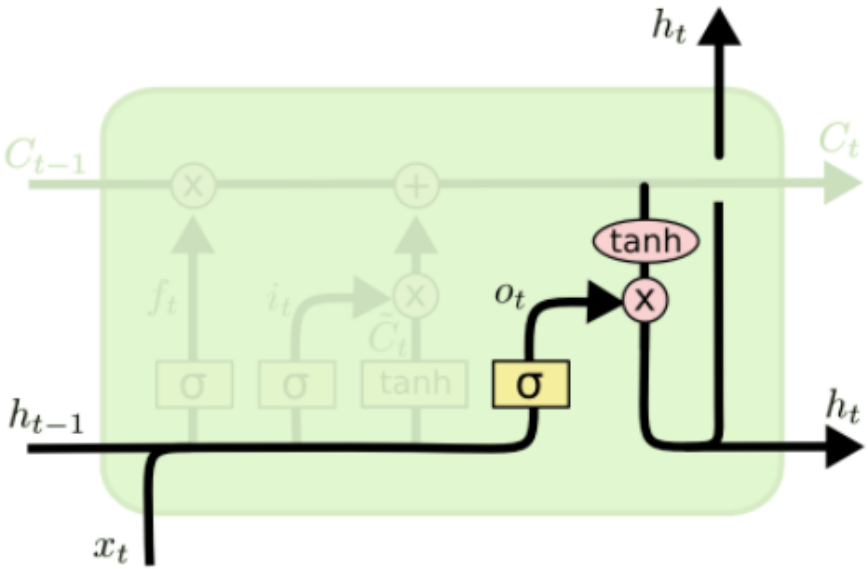
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM Architektur - Update Mechanism



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM Architektur - Output Mechanism



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Implementation

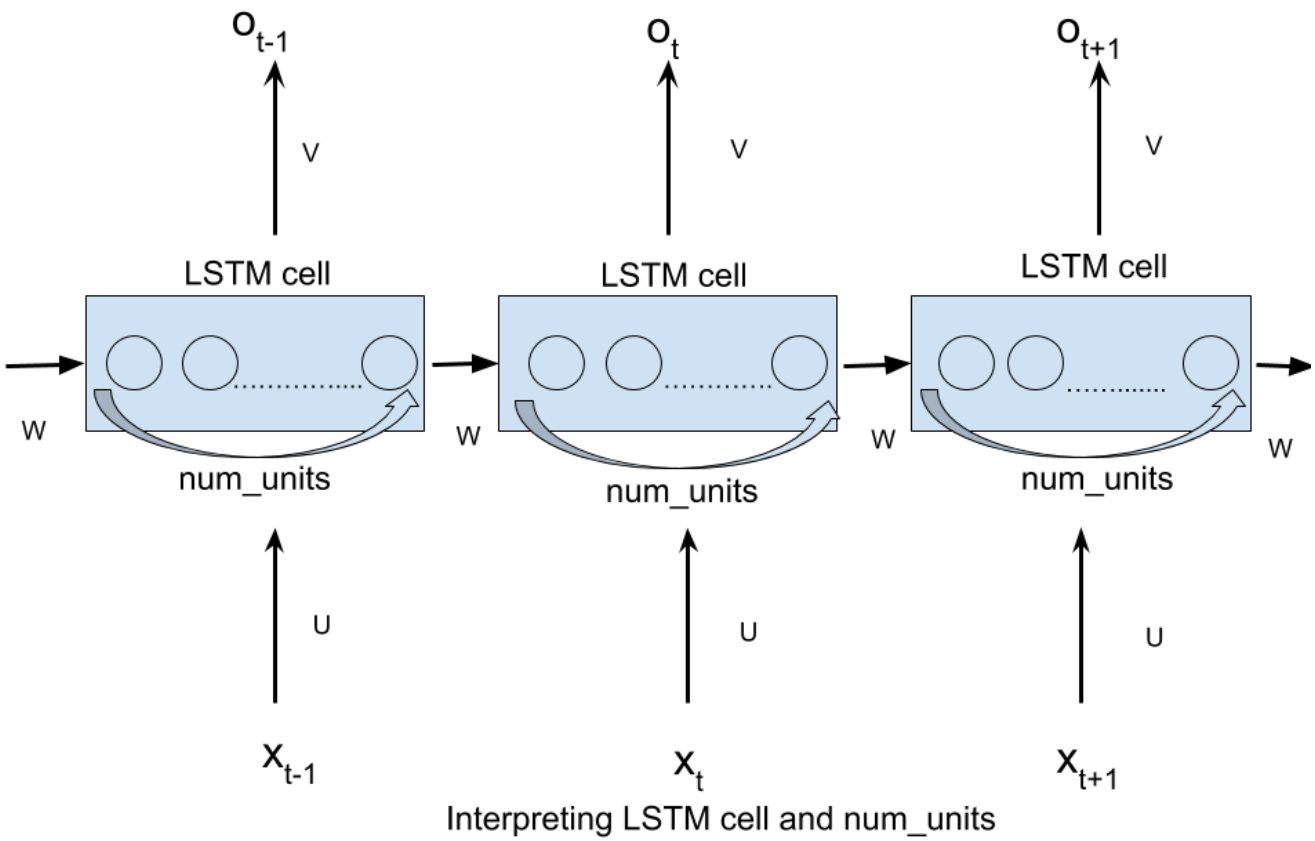
- Python, R, Java
- Python, Keras mit Tensorflow backend
  - Python 3.7.6
  - Tensorflow 2.3.0
  - Keras 2.4.3

Implementation - num_units

```
import ...

# define model
model = Sequential()
model.add(LSTM(units = 50, input_shape=(3, 1)))
...
```

Implementation - num_units



Interpreting LSTM cell and num_units

Nachfolgende Bilder von: https://jasdeep06.github.io/posts/Understanding-LSTM-in-Tensorflow-MNIST/ [aufgerufen 4 Oktober 2020].

Implementation - Loss und Optimizer

```
model.compile(loss='mse', optimizer='adam')
```

- MSE $= \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$
- Adam
  - De facto Standard
  - Adaptive Learning Rate
  - Momentun um lokaler Minimas zu entfliehen

Implementation

LSTM Vor- und Nachteile

- Vorteile
  - Kontext
  - Kaum Feature Engineering
  - Anwendbar auf verschiedene Probleme
- Nachteile
  - Tendenziell viele Parameter
    - Lange Trainingszeit
    - Tendiert zun Overfitten
  - Erklärbarkeit
  - kein Konfidenzintervall

Ausblick

- GRU
- Attention
- CNN - LSTM (ConvLSTM)
- xAI

Referenzen

T. Yiu

Understanding Neural Networks

https://towardsdatascience.com/understanding-neural-networks-19020b758230 Aufgerufen am 04.11.2020

H. Lohninger

Zeitreihen - Vorhersage

*http://www.statistics4u.info/fundstat$_g$erm/cc$_t$imeser$_f$orecast.htmlAufgerufenam*04.11

Y. Fenjiro

Time Series for Business: A general introduction

https://medium.com/@fenjiro/time-series-for-business-a-general-introduction-50968346e660 Aufgerufen am 04.11.2020

References

C. Olah (colah)

Understanding LSTM Networks – colah's blog (August 2015)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/ Aufgerufen am 04.11.2020

jasdeep06

Understanding LSTM in Tensorflow

https://jasdeep06.github.io/posts/Understanding-LSTM-in-Tensorflow-MNIST/ Aufgerufen am 04.11.2020

J. Hennebert, M. Melchior

MSE TSM Deep Learning (HS 2019)

Vorlesung 12: Recurrent Neural Networks

www.github.com/AntonioRosolia/Methodenseminar

Fragen