./Bilder/Bild1.jpeg0.4

Zeitreihenvorhersage

am Beispiel der Sonnenflecken

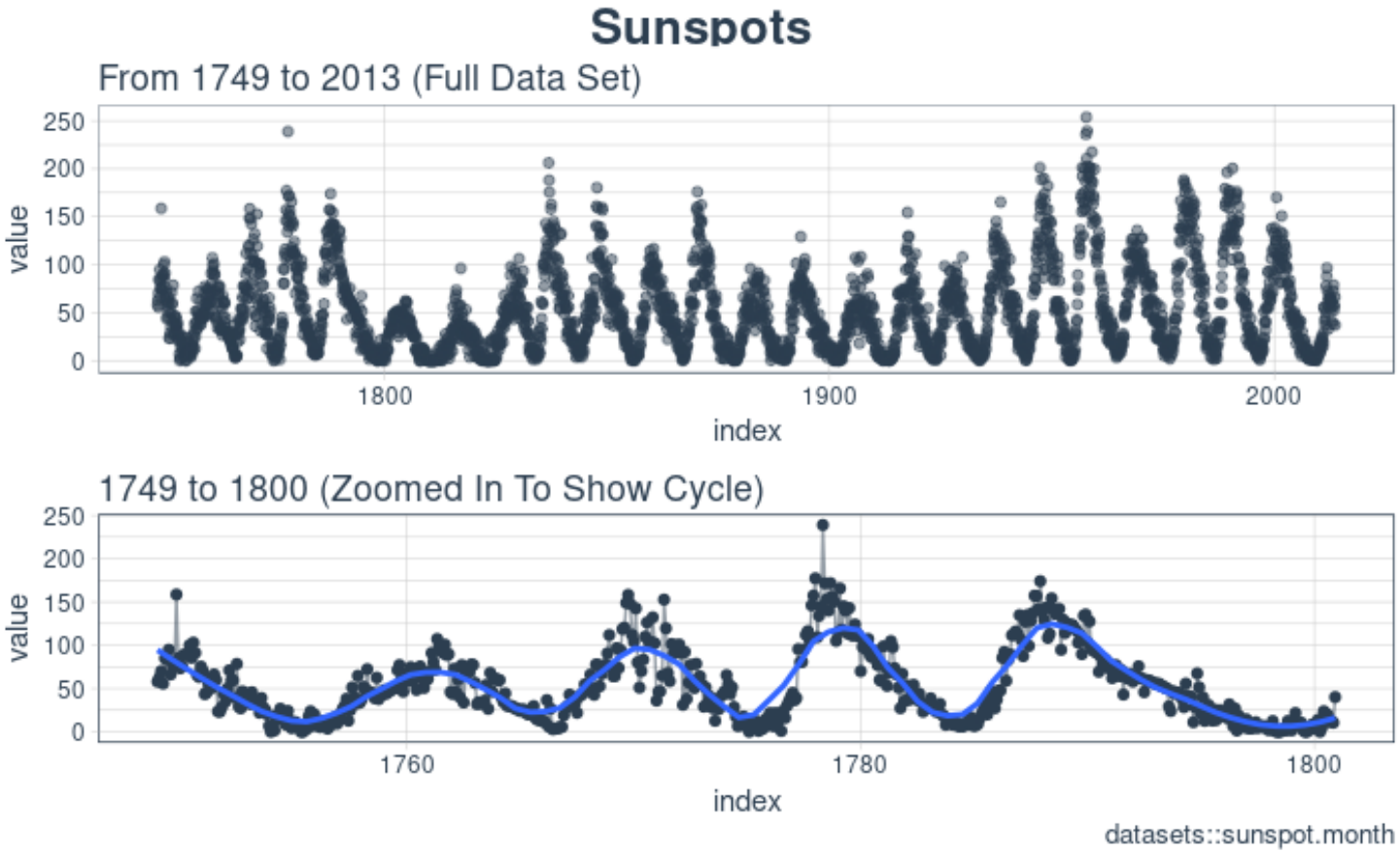
****** (anonymisiert)

26. April 2024

[text and total]

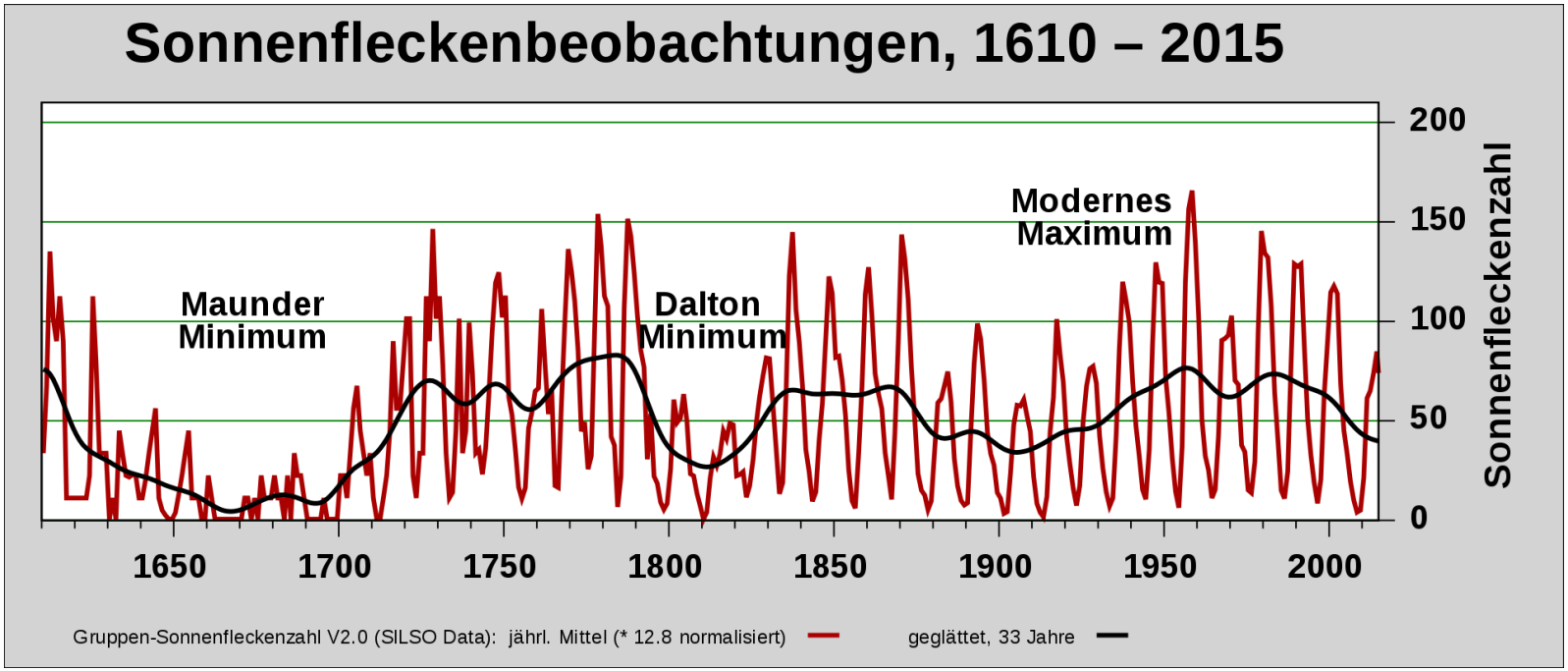
Messreihe

Daten von der SIDC



https://blogs.rstudio.com/ai/posts/2018-06-25-sunspots-lstm/

Einfluss auf Erde



- Sonnenstrahlung schwankt um ca. 0,1%
- Auswirkung auf Temp. ca. 0,1 °C
- Koronalereignisse, wie 1859, Nordamerika

Inhalt

Teil

Teil

Datenaufbereitung 1

Ziel: Regressionsmodell f welches aus x eine Vorhersage $y = f(x)$ liefert.

$\Rightarrow$ Fragen:

- Klasse der Modelle f?
- Länge von x?
- Länge von y?
- Mehrere Zeitreihen zum Trainieren und Validieren?

Datenaufbereitung 1

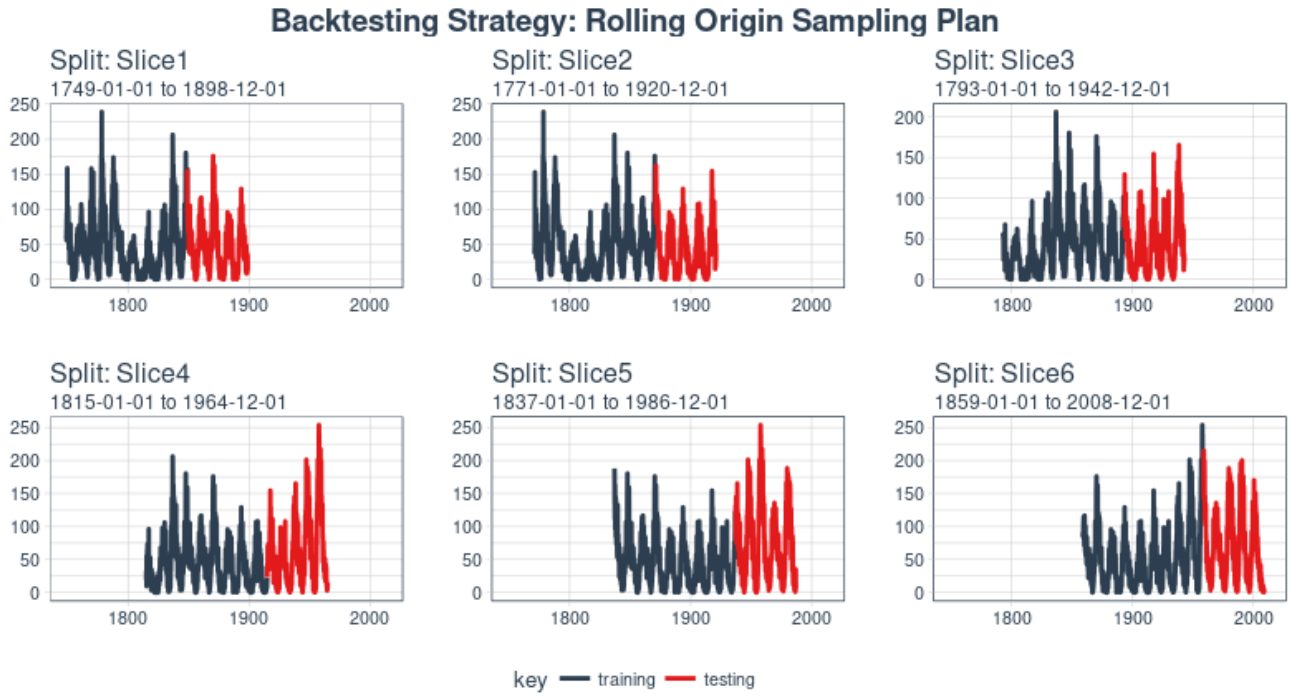
Ziel: Regressionsmodell f welches aus x eine Vorhersage $y = f(x)$ liefert.

$\Rightarrow$ Fragen:

- Klasse der Modelle f? $\rightarrow$ RNN, LM, RW, SARIMA, Markov, ...
- Länge von x? $\rightarrow nt_x$
- Länge von y? $\rightarrow nt_y$
- Mehrere Zeitreihen zum Trainieren und Validieren? $\rightarrow$ Rückvergleich

Datenaufbereitung 1

1.1 Aufteilung zur Validierung: verschobene Zeitfenster



https://blogs.rstudio.com/ai/posts/2018-06-25-sunspots-lstm/

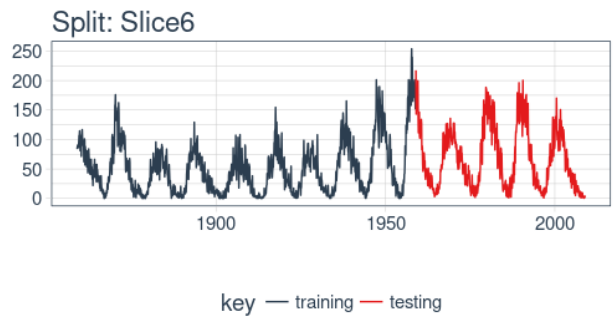
- blau: Training, rot: Testen
- statt typischer Kreuzvalidierung: 6 kleine Fenster

Datenaufbereitung 1

1.2 Weitere Aufteilung zum Trainieren und Testen

Erzeuge x- und y- Werte nach Schema:

$$1, 2, 3, 4, 5, 6, 7 \quad \rightarrow \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix}, \begin{pmatrix} 4 & 5 \\ 5 & 6 \\ 6 & 7 \end{pmatrix}$$



https://blogs.rstudio.com/ai/posts/2018-06-25-sunspots-lstm/

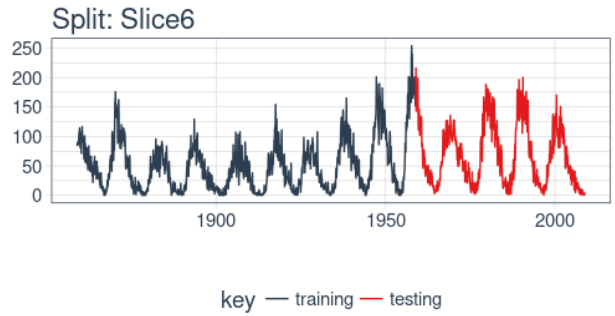
Datenaufbereitung 1

1.2 Genauer:

- $\underbrace{b_1, b_2, \ldots}_{nt_x} \underbrace{\ldots\ldots\ldots}_{nt_y} \ldots$ wird zu

$$X_{train} \cong \begin{pmatrix} b_1 & \ldots & b_{nt_x} \\ b_2 & \ldots & b_{nt_x+1} \\ \vdots & & \vdots \end{pmatrix}, Y_{train} \cong \begin{pmatrix} b_{nt_x+1} & \ldots & b_{nt_x+nt_y} \\ b_{nt_x+2} & \ldots & b_{nt_x+nt_y} \\ \vdots & & \vdots \end{pmatrix}$$

- $r_1, r_2, \ldots$ wird zu X_{test}, Y_{test}



https://blogs.rstudio.com/ai/posts/2018-06-25-sunspots-lstm/

Datenaufbereitung 2

Möglichkeiten:

- Skalierung und Zentrierung (z.B. wichtig für Gradientenverfahren)
- Differenzieren (kann Stationariät erzeugen)[1]

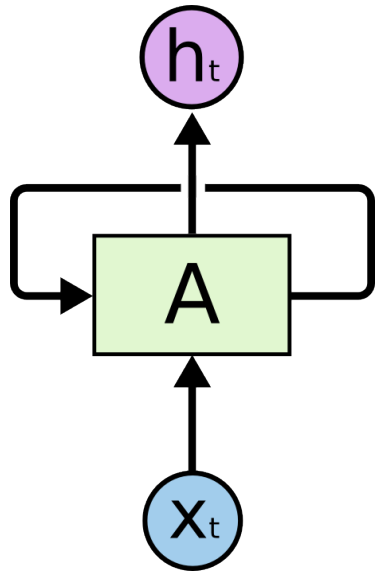
[1] kpss.test(sunpots.month)$statistic = 1.1505, also Stationarität verwerfen! Aber kpss.test(diff(sunspot.month)) = 0.007631239.

Teil

Teil

Rekurrente Netze

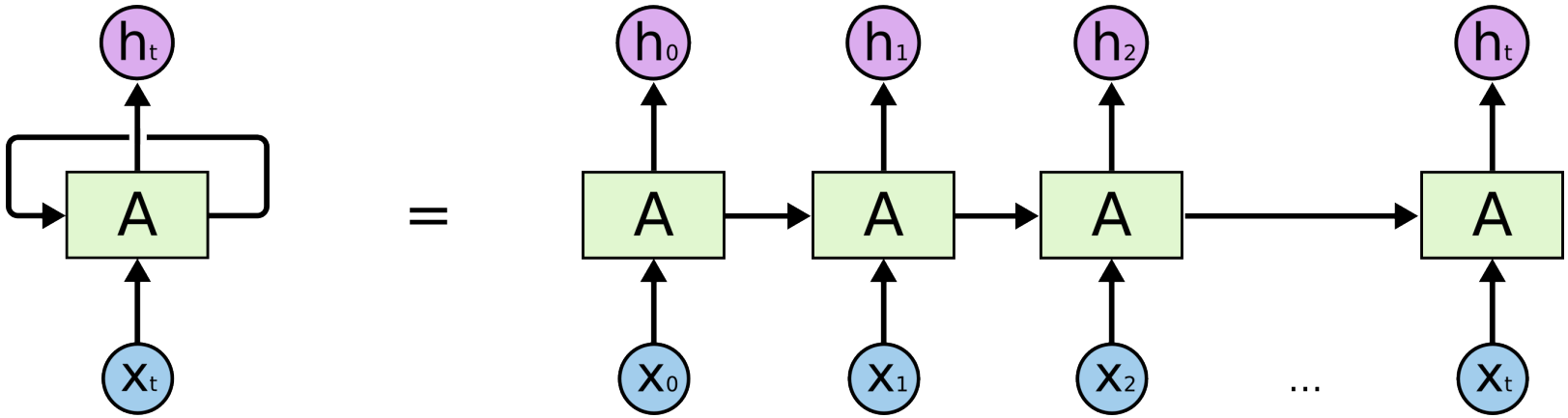
Idee: Informationen merken und wiederverwenden



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrente Netze

- Training als normales mehrschichtiges Netz

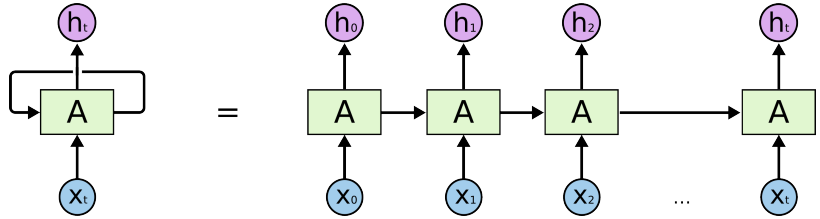


https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- O.B.d.A. h_{t-1} und Gedächtnis identitsch
- Problem: Verschwindende/Eplodierende Gradienten

Rekurrente Netze

Verschwindende Gradienten



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Hier Annahme: $h_t = \sigma(wh_{t-1} + ux_t)$
- Gradienten berechnen:

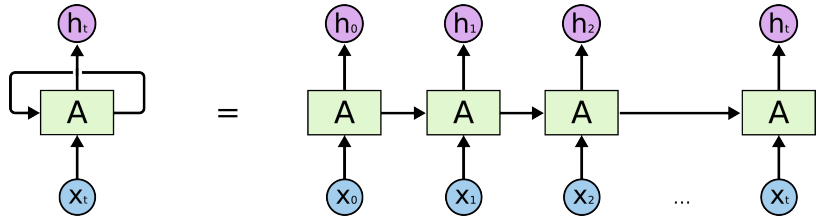
$$\frac{\partial h_3}{\partial w} = \frac{\partial \sigma(wh_2 + ux_3)}{\partial w} = \sigma' \cdot \left[h_2 + w \frac{\partial h_2}{\partial w} \right] = \sigma' h_2 + \sigma' w \frac{\partial \sigma(wh_1 + ux_2)}{\partial w}$$

$$= \sigma' h_2 + \sigma' w \sigma' \cdot \left[h_1 + w \frac{\partial h_1}{\partial w} \right]$$

$$= \sigma' h_2 + \sigma' \sigma' w h_1 + \sigma' \sigma' w^2 \frac{\partial h_1}{\partial w}$$

Rekurrente Netze

Verschwindende/Explodierende Gradienten



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Hier Annahme: $h_t = \sigma(wh_{t-1} + ux_t)$
- Gradienten berechnen:

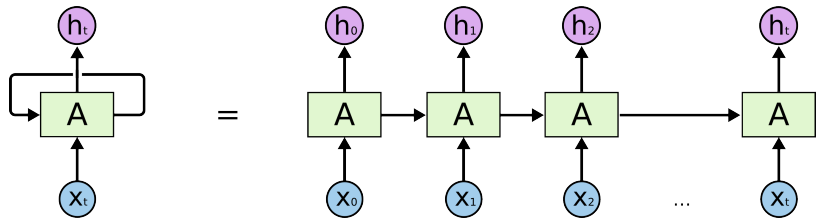
$$\frac{\partial h_3}{\partial w} = \sigma' h_2 + \sigma'\sigma' w h_1 + \sigma'\sigma' w^2 \frac{\partial h_1}{\partial w}$$

- $h_2 = h_2(x_2)$ und $h_1 = h_1(x_1)$
- Wenn $|\sigma' w| < 1$, dann x_1 viel weniger Einfluss als x_2!

Rekurrente Netze

Verschwindende/Explodierende Gradienten

Lösungsideen:

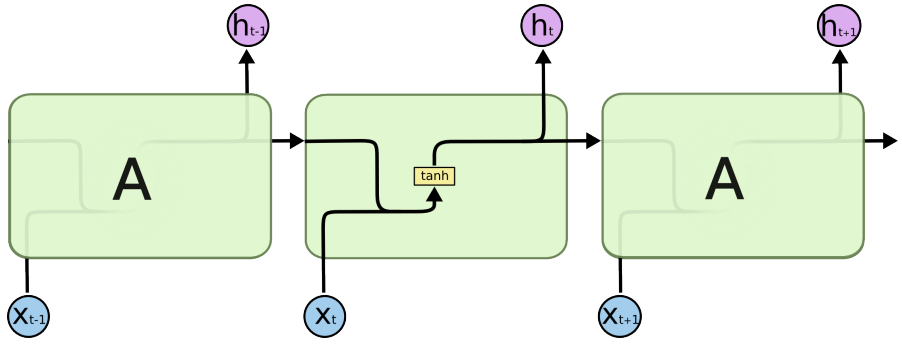


https://colah.github.io/posts/2015-08-Understanding-LSTMs/

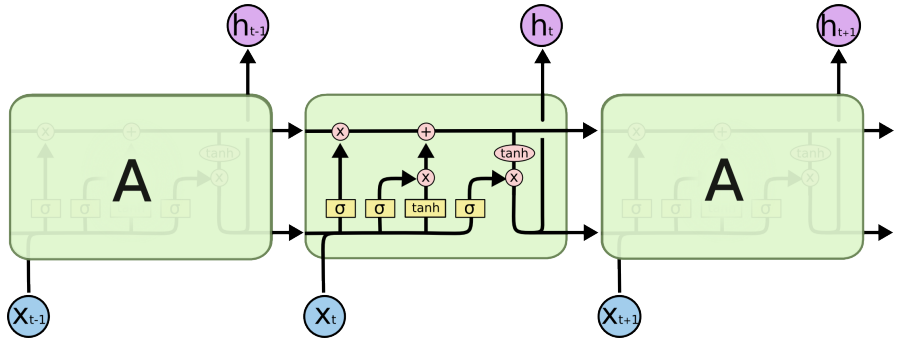
- Andere Aktivierungsfunktionen (z.B. ”Relu”)
- LSTM, . . .

LSTM-Netze

”Normales”RNN:



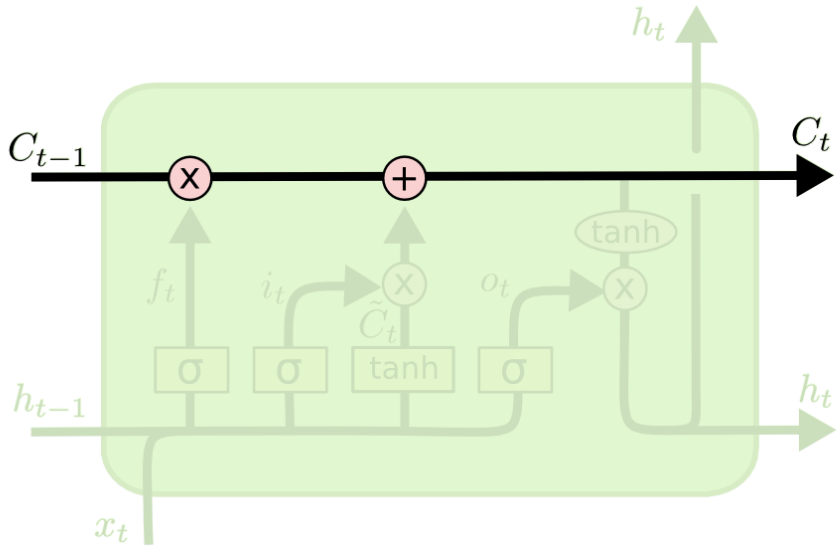
LSTM-Netz („Long Short Term Memory“):



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM-Netze

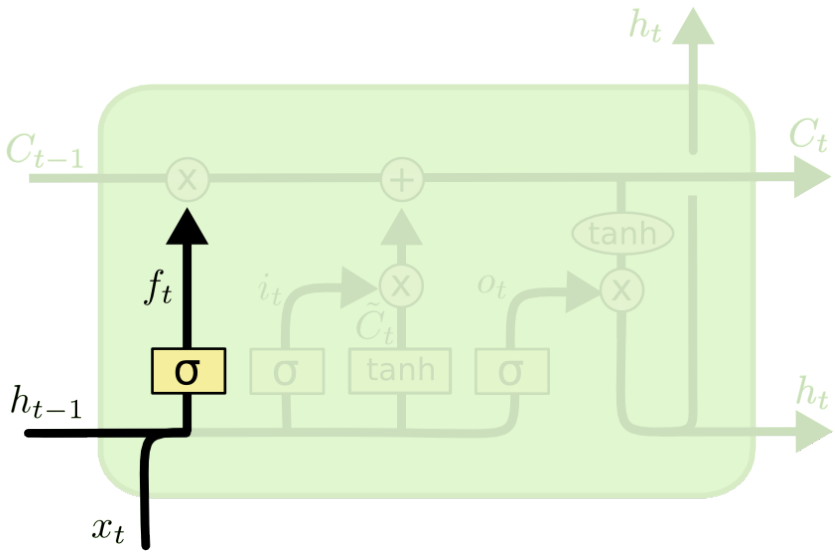
Memory



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM-Netze

Steuerung Forget Gate

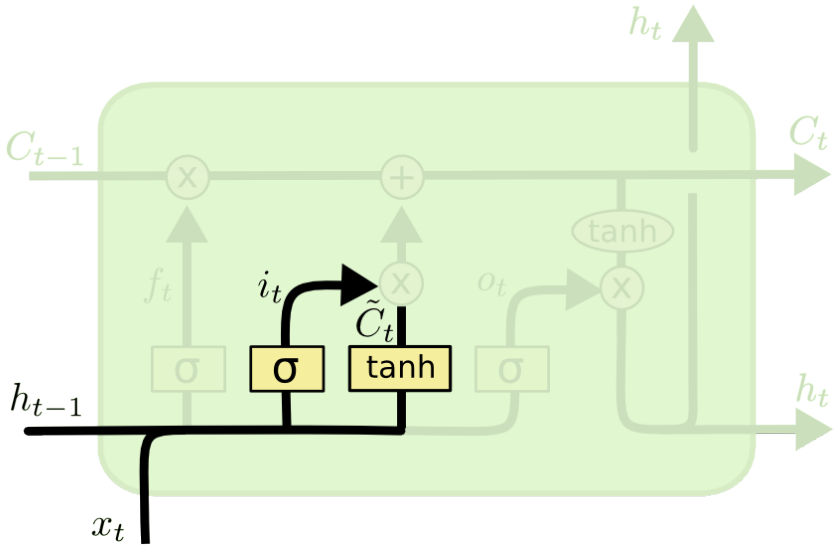


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM-Netze

Steuerung Input Gate



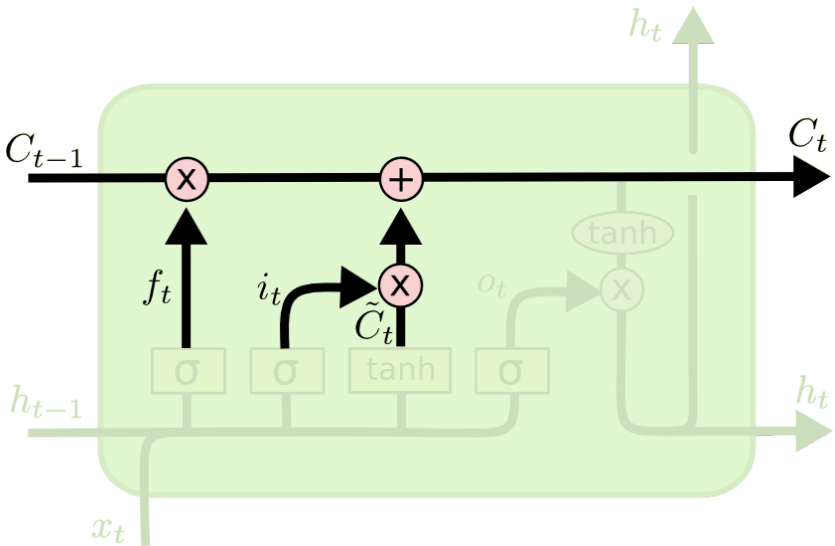
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM-Netze

Update Memory

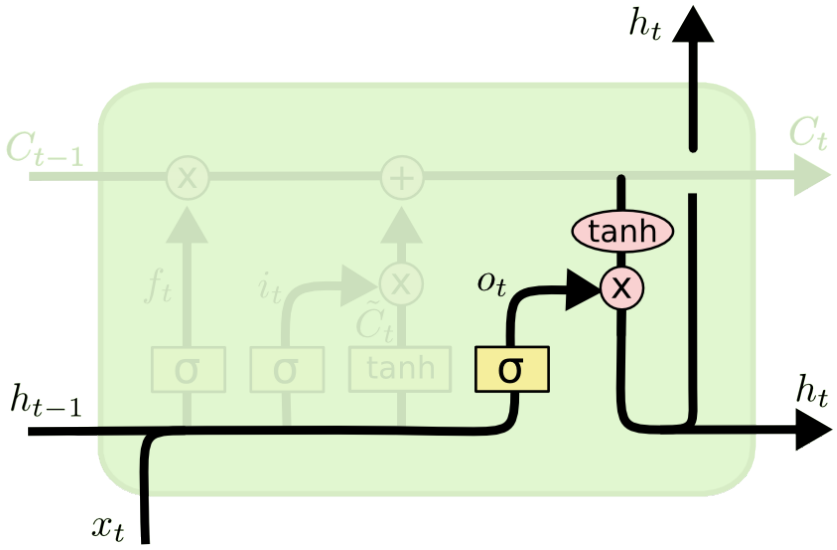


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM-Netze

Output Gate



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

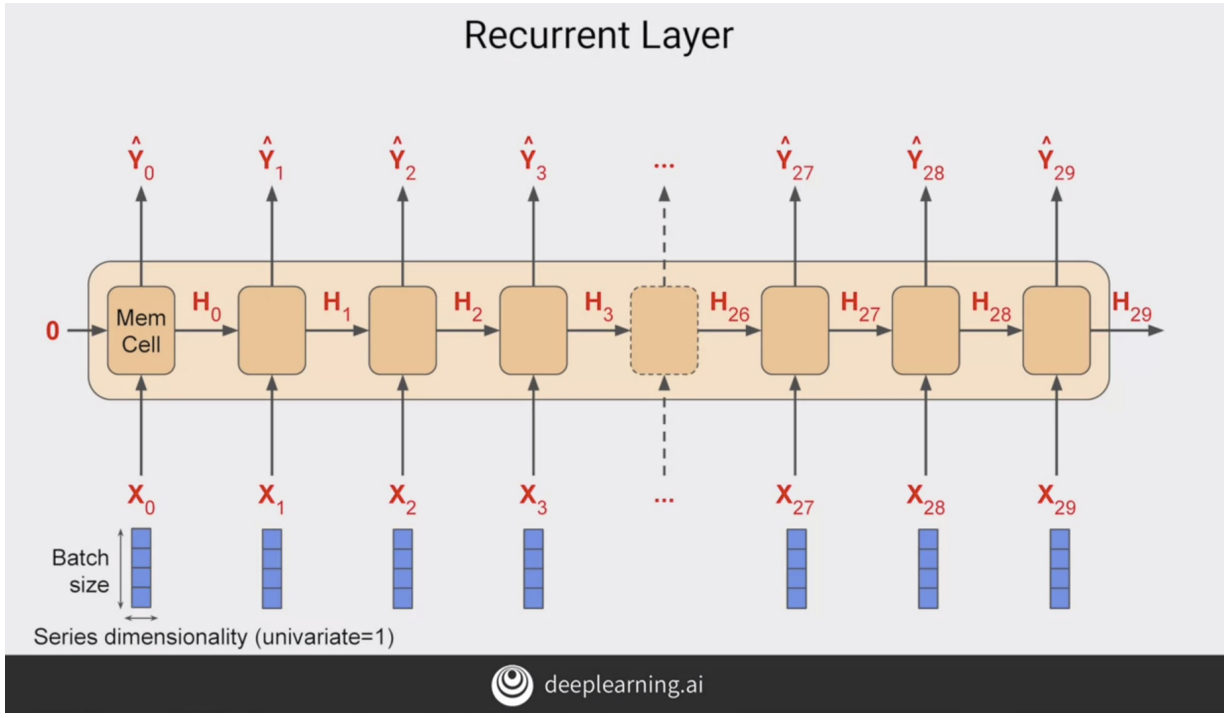
$$h_t = o_t * \tanh\left(C_t\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Teil

Teil

Skizze für unser Netz



https://i.stack.imgur.com/nSExT.png

n_timesteps_x = n_timesteps_y = 29
batch_size = 4
return_sequences = TRUE
Wichtig: Nur ein Neuron mit Speichergröße n_units! Und die nächste Schicht merkt nichts von den Zeitschritten.

Quellenangabe

- Auf den folgenden Folien: Code von blogs.rstudio.com/ai/posts/2018-06-25-sunspots-lstm/
- Wahl der Parameter:

```
n_timesteps_x <- 12
n_timeststeps_y <- 12
batch_size <- 10
```

Abgespeckter Code

```
# Tensorflow einbinden
library(keras)

# Fundament:
modell <- model_sequential()

# Zwei Schichten hinzufuegen:
layer_lstm(modell,
           units = 128,
           batch_input_shape = c(batch_size,
                                 n_timesteps_x,
                                 n_features),
           return_sequences = TRUE
)

time_distributed(layer_dense(modell, units = 1))
```

LSTM und Tensorflow

```
# Deswegen batch_size fix:
compile(modell,
        loss = "logcosh",
        optimizer = "sgd",
        metrics = list("mean_squared_error")
)

# Trainieren
history <- fit(modell,
        x = X_train # dim = c(770, 12, 1),
        y = Y_train # dim = c(770, 12, 1),
        batch_size = batch_size, # 10,
        epochs = 100
)
```

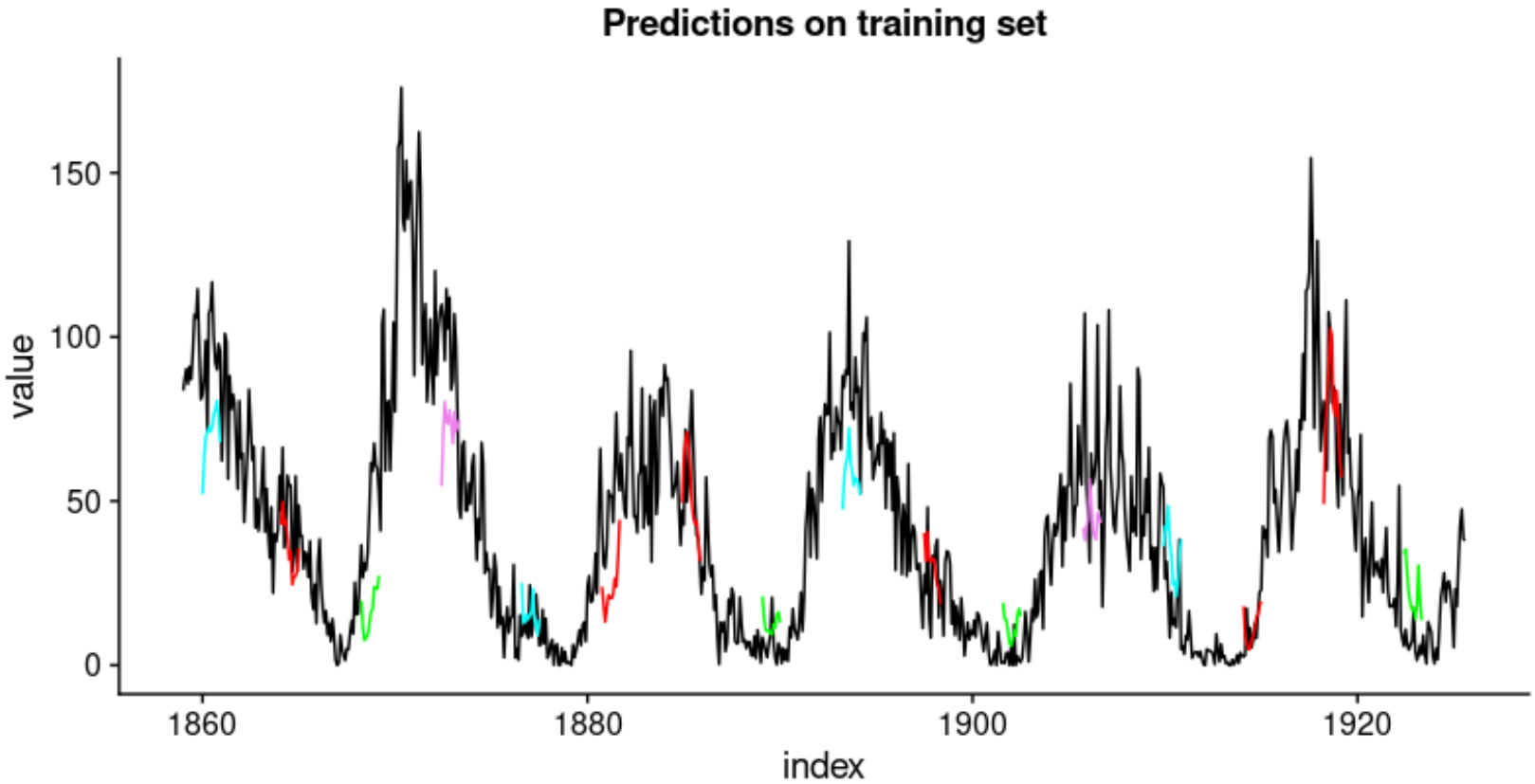
Abgespeckter Code

```
# Vorhersage fuer Testdatensatz (Validierung)
pred_test <- predict(modell, X_test, batch_size = 10)
```

Teil

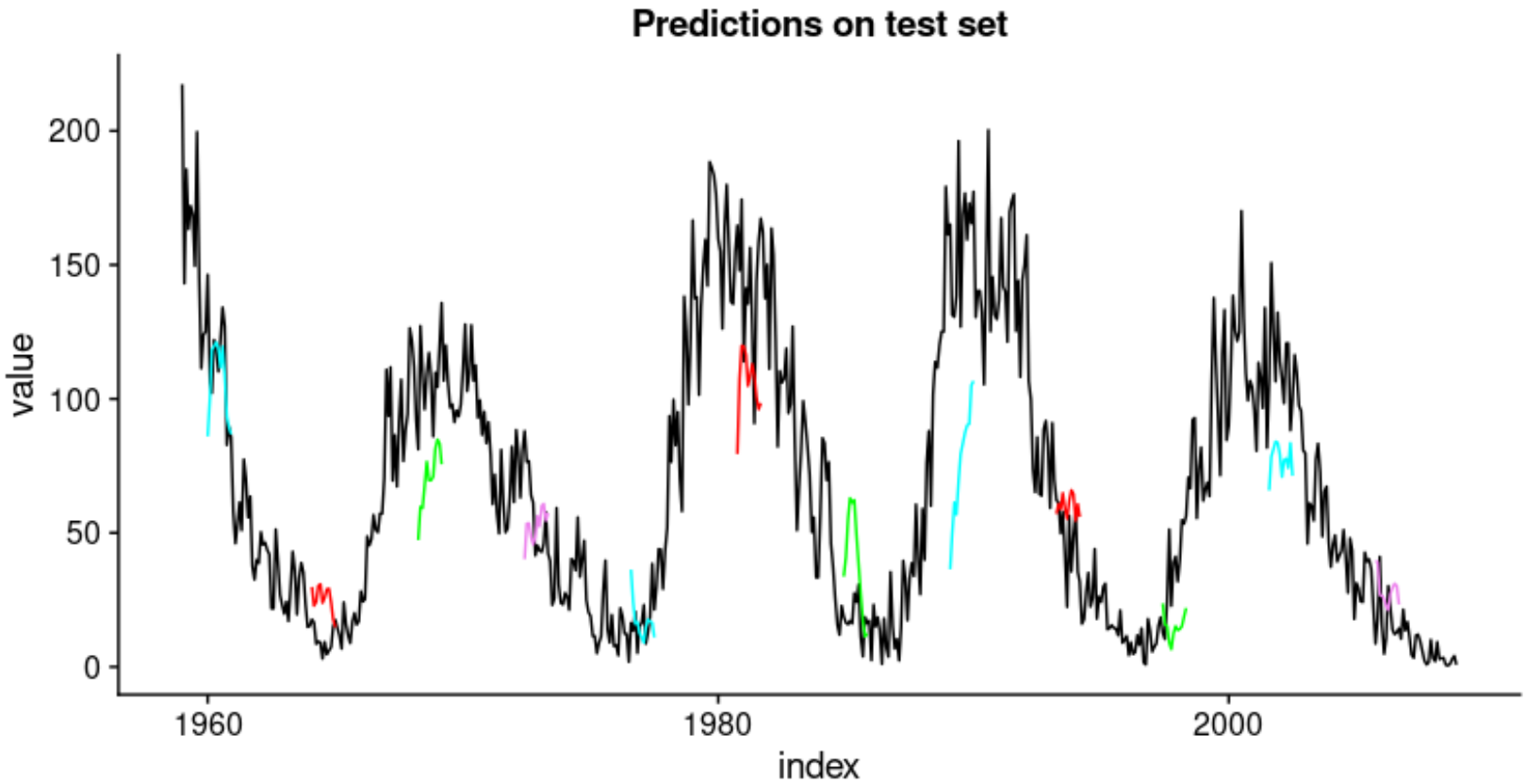
Teil

Fit der Trainingsdaten



RMSE = 21.01

Vorhersage mit Testdaten



RMSE = 31.32

Teil

Teil

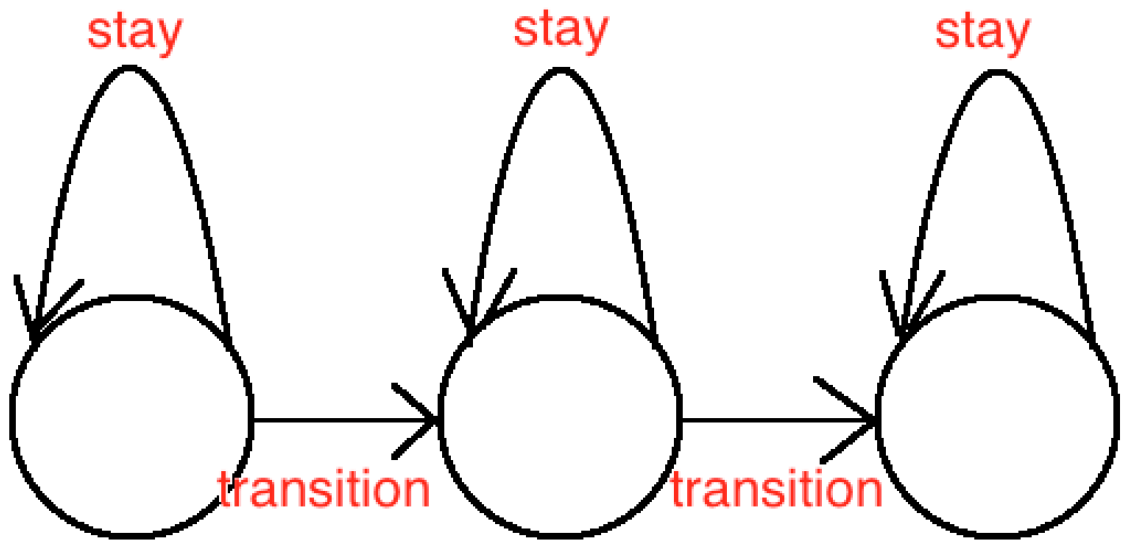
(S)ARIMA und Ähnliches

Einfachster Modellansatz:

$$s_n = \sum_{k=1}^{p} a_k s_{n-k}$$

Erweiterungen: Saisonale Effekte, Differentiation/Integration, ...

Markov-Methoden



https://cdn-images-1.medium.com/max/1600/1*UsePQeFxlK67cGmluHzqsA.png

Paper zu dem Thema: Novitasari, Ardhiyah and Widodo, Flare Identification by Forecasting Sunspot Numbers Using Fuzzy Time Series Markov Chain Model

Lineares Modell und Regressionswald

Idee: Erzeuge 12 selbstständige Modelle[2]

$$f_i(s_1, \ldots, s_{12}) \approx s_{12+i}, \qquad i = 1, \ldots 12$$

Ergebnis:

- RMSE des LM: 21.95295
- RMSE des RW: 24.382

(Bilder siehe Markdown)

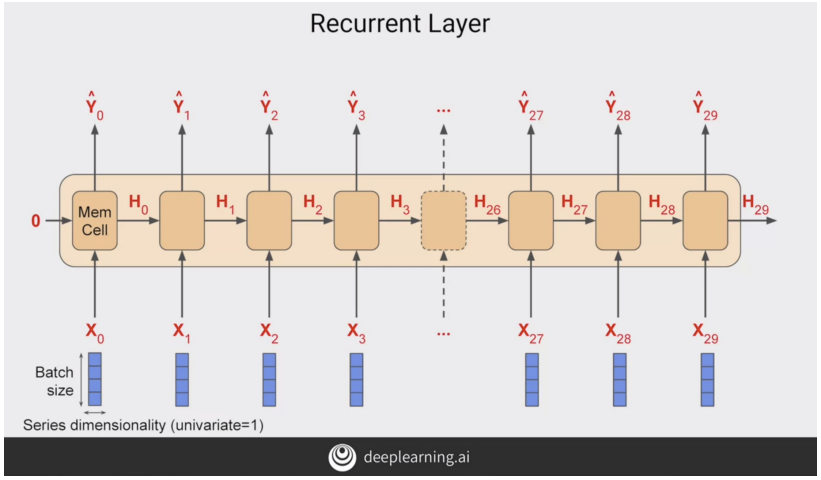
Zum Vergleich RMSE des LSTM 31.32

[2] https://www.r-bloggers.com/2019/09/time-series-forecasting-with-random-forest/

Teil

Teil

Nur die letzte Vorhersage betrachten

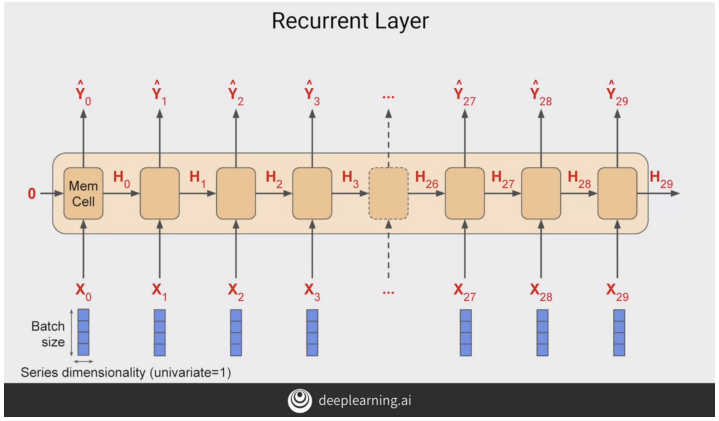


https://i.stack.imgur.com/nSExT.png

Bisher musste das Netz die ersten Vorhersagen, ohne Kenntnis der Vergangenheit treffen!
⇒ Vergleiche RMSE nur für letzte Vorhersagen:

- LM: 33.22806
- RW: 37.1156
- LSTM: 39.03997

Einstufiges Netz iterieren



https://i.stack.imgur.com/nSExT.png

Idee:

$$f_{LSTM}(s_1, s_2, \ldots s_{12}) =: \hat{s}_{13}$$

$$\ldots$$

$$f_{LSTM}(s_{12}, \hat{s}_{13} \ldots, \hat{s}_{23}) =: \hat{s}_{24}$$

$\Rightarrow$ RMSE des LSTM: 32.3842
(Erinnerung: LM: 21.95295, RW: 24.382)

Weitere Verbesserungsideen

- Verlängere Zeithorizont
- Validierung mit allen sechs Zeitfenstern
- Physikalisch sinnvolles Modell anpassen (z.B. mit Prophet? https://peerj.com/preprints/3190/)
- Kombination von ARIMA, ETS, SVM: https://link.springer.com/content/pdf/10.1007/s11207-020-01757-2.pdf

Teil

Teil

Schlussworte

- LM und RW wesentlich besser.[3]
- Hauptaufwand: Datenaufbereitung, Parametertuning
- Vorsicht mit Posts zum Thema ML!

[3] Siehe auch Präsentation von J. Nebel und https://iopscience.iop.org/article/10.1088/1742-6596/1231/1/012022/pdf

Vielen Dank für die Aufmerksamkeit!