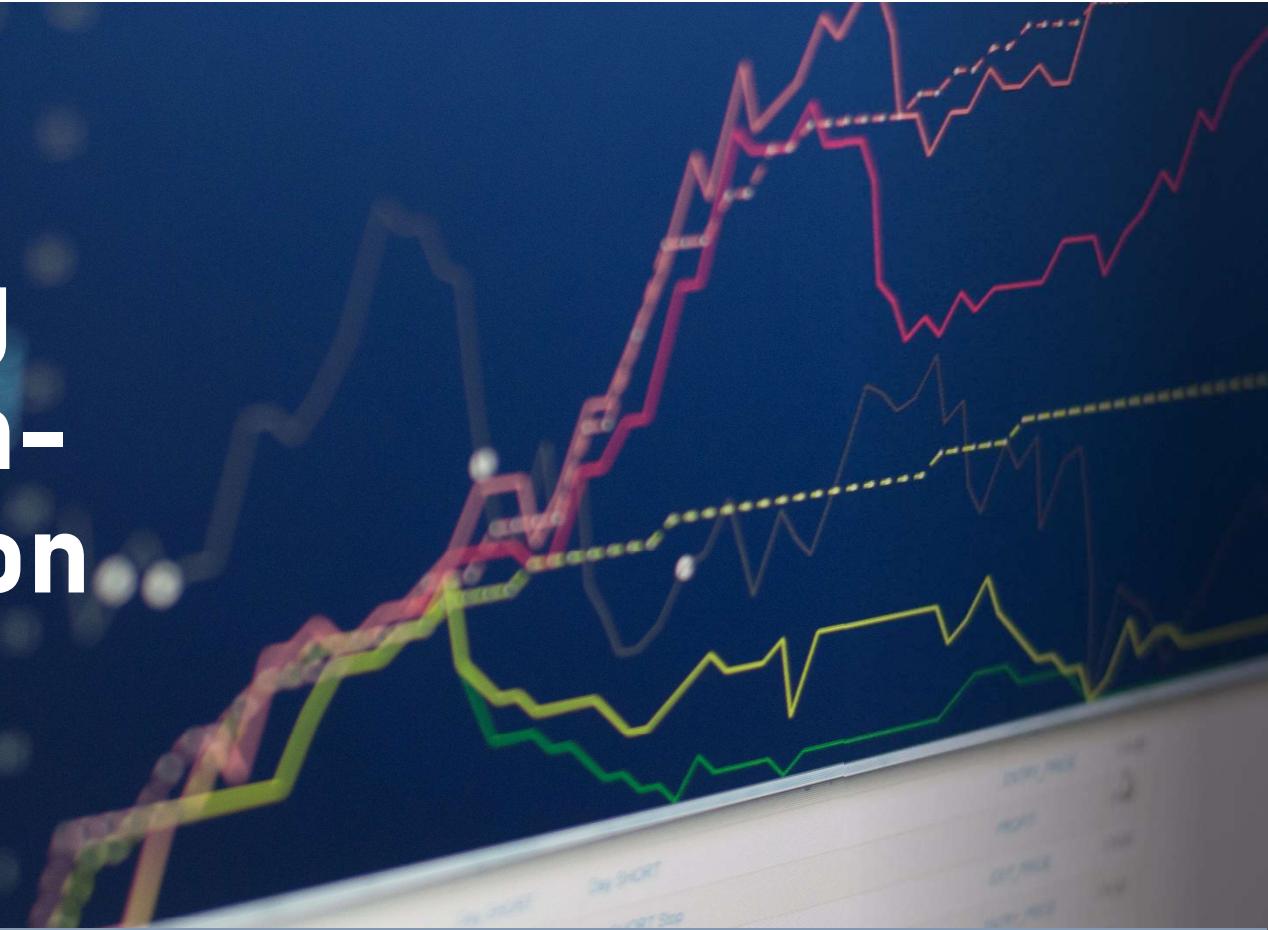


Analyse und Visualisierung von Zeitreihen- Daten in Python



Hmetrica



Analyse und Visualisierung von Zeitreihen-Daten in Python

Tag 2

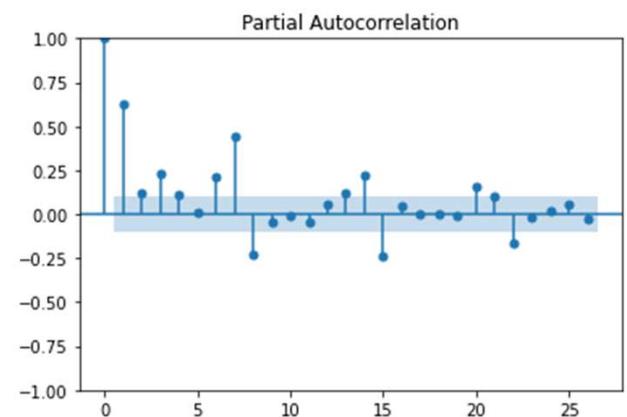
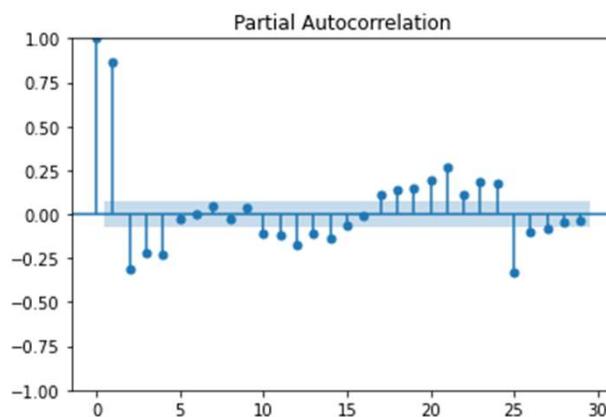
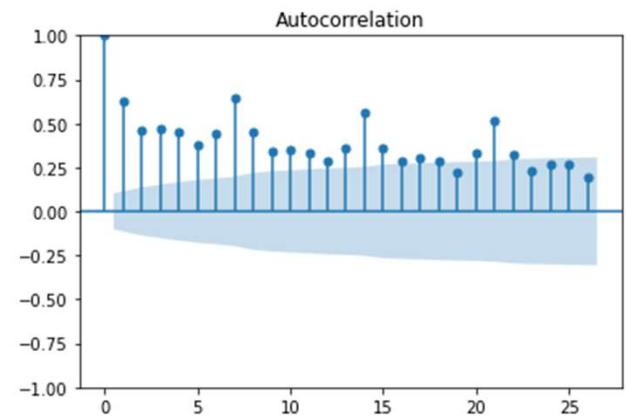
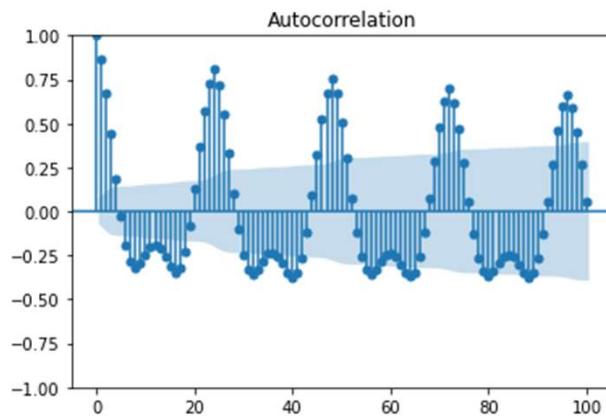


Einwürfe von gestern



Resampling

Tage statt Stunden



Datenvorverarbeitung

Datenbereinigung:

- Fehlende Werte: Identifizieren und behandeln fehlender Werte: z.B. Auffüllen mit dem vorherigen/nächsten Wert, lineare Interpolation o.ä.
- Ausreißer: Statistische Ausreißer identifizierung und behandeln

Datentransformation:

- Skalierung: Normalisieren / standardisieren der Daten, besonders bei maschinellem Lernen (ML ist empfindlich auf unterschiedliche Wertebereiche)
- Transformation: Bei Bedarf Transformationen wie Logarithmieren oder Differenzbildung, um die Stationarität zu verbessern oder die Varianz zu stabilisieren.

Datenvorverarbeitung

Zeitreihenspezifische Aspekte:

- Saisonalität und Trend: Saisonale Muster und Trends modellieren
- Stationarität: Auf Stationarität prüfen, da viele Zeitreihenanalyse-Methoden dies voraussetzen

Feature-Engineering:

- Zeitbasierte Features: Aus Zeitstempel zusätzliche Features wie Wochentag, Monat, Jahreszeit, Feiertage usw. erzeugen
- Rolling-Statistiken: z.B. gleitende Durchschnitte um Trends hervorzuheben

Datenaufteilung:

- Chronologische Aufteilung: Daten in Trainings- und Testsets auf einer zeitlichen Basis aufteilen, nicht zufällig!

Umgang mit Stationarität

De-Trending

- Genutzt für Nicht-Stationarität aufgrund von Trendkomponente
- Einfaches Bsp:

$$T_t = \alpha + \beta t + Z_t$$

- Lineare Regression

$$T_t = \alpha + \beta t + Z_t$$

- Zeit t als Prädiktor, Zeitreihendaten T_t als Zielvariable

- De-trendete Daten T'_t

$$T'_t = T_t - \hat{T}_t = T_t - (\alpha + \beta t)$$

- Für einfaches Modell: $T'_t = T_t - (\alpha + \beta t) = Z_t$

Achtung: Komplexere Modelle bestehen nach dem De-trenden nicht nur noch aus weißem Rauschen

Umgang mit Stationarität

Erste Differenzen

- Genutzt für Nicht-Stationarität aufgrund von Einheitswurzel
- Einfaches Modell mit Einheitswurzel:

$$X_t = \phi X_{t-1} + Z_t$$

mit ϕ sehr nahe 1

- Erste Differenz für Zeitpunkt t :

$$\Delta Y_t = Y_t - Y_{t-1}$$

- Für einfaches Modell:

$$\Delta X_t = (\phi X_{t-1} + Z_t) - X_{t-1} = (\phi - 1)X_{t-1} + Z_t \approx Z_t$$

- Auch für einfaches Trendmodell:

$$\Delta T_t = (\alpha + \beta t + Z_t) - (\alpha + \beta(t-1) + Z_{t-1}) = \beta + Z_t - Z_{t-1}$$

Achtung: Komplexere Modelle bestehen nach dem Differenzieren nicht nur noch aus weißem Rauschen

Inhalte – Was haben wir vor?

Tag 1

1. Einführung in Zeitreihendaten in Python
2. Zeitreihen und ihre Merkmale visualisieren
3. Zeitreihen vorhersagen (Statistik I): Exponentielle Glättung und Holt-Winters
4. Zeitreihen vorhersagen (Statistik II): ARIMA-Modelle

Tag 2

5. Einblick in andere Zeitreihenmodelle
6. Machine Learning für Zeitreihen: Überblick, Vorbereitung und Vorhersagen
7. Machine Learning für Zeitreihen: Clustering und Klassifikation
8. Deep Learning für Zeitreihen

Einblick in andere Zeitreihenmodelle

Session 5 (Dienstag 09:15 – 10:45)



Einblick in andere Zeitreihenmodelle

1. Frequenzanalyse
2. Zustandsraumdarstellung (State-space models)
3. Additives Regressionsmodell (Facebook Prophet)

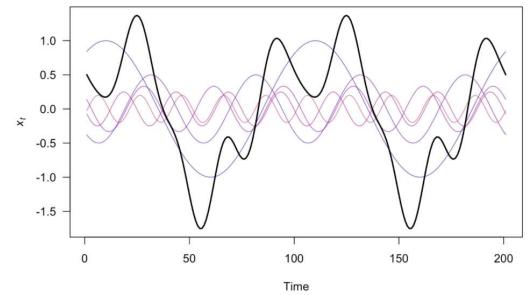
Frequenzanalyse

- Statt Änderungen der Amplitude *mit der Zeit*: Amplitudenänderung *mit der Frequenz*
- Die Fourier-Transformation ist eine der häufigsten Methoden zur Analyse von Frequenzkomponenten in Zeitreihen.
- Sie zerlegt die Zeitreihe in verschiedene harmonische Sinus- und Cosinusfunktionen, die verschiedene Frequenzen repräsentieren.
- Diese Frequenzkomponenten können dann analysiert werden, um Einblicke in die strukturellen Muster der Zeitreihe zu gewinnen.

Beispiel

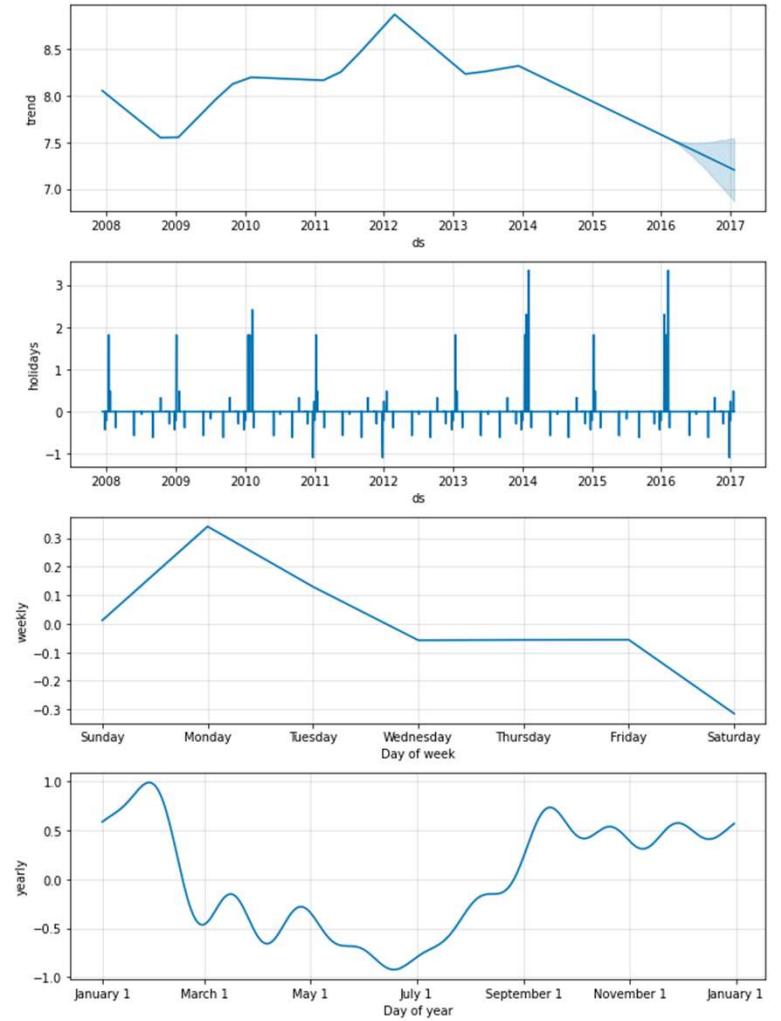
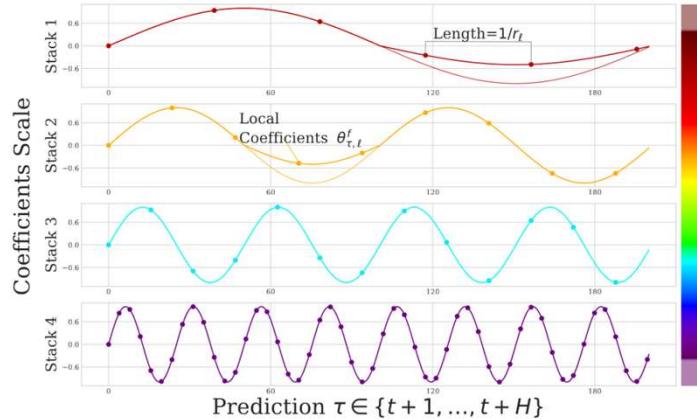
Unsicherheit in der Erzeugung von Windstrom durch Abschneiden der Hochfrequenzkomponente zuverlässiger modellieren (Oh & Son, 2018)

$$f(t) = \sum_{k=1}^5 \frac{1}{k} \sin(2\pi kt + k^2)$$



Frequenzanalyse

- Funktioniert gut für Identifizierung von periodischen, durch Rauschen verfälschten Signalen.
- Inkonsistenter Schätzer für die meisten realen Datensätze
- Aber: **Saisonale Komponenten** lassen sich so gut modellieren



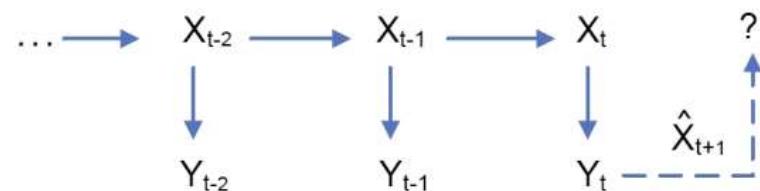
State-Space Models

Zustandsraumdarstellung

- Aus den Ingenieurwissenschaften: alle Beziehungen der Eingangs-, Ausgangs- und Zustandsgrößen in Matrizen und Vektoren dargestellt
- **Kalman-Filter:** Algorithmus mit Prädiktor-Korrektor-Struktur

Beispiel

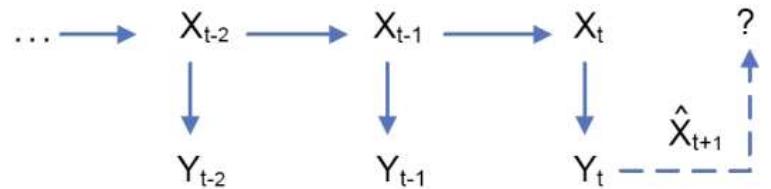
Sie interessiert die Innen-Raumtemperatur X_t , aber: Messung Y_t an einer Ecke des Raums mit einem Sensor



State-Space-Models

Zustandsraummodell (einfachste Form)

- Zustandsgleichung $X_{t+1} = F_t X_t + W_t$
- Beobachtungsgleichung $Y_t = G_t X_t + V_t$
- ($W_t \sim IID(0, Q_t)$ und $V_t \sim IID(0, R_t)$ weißes Rauschen)



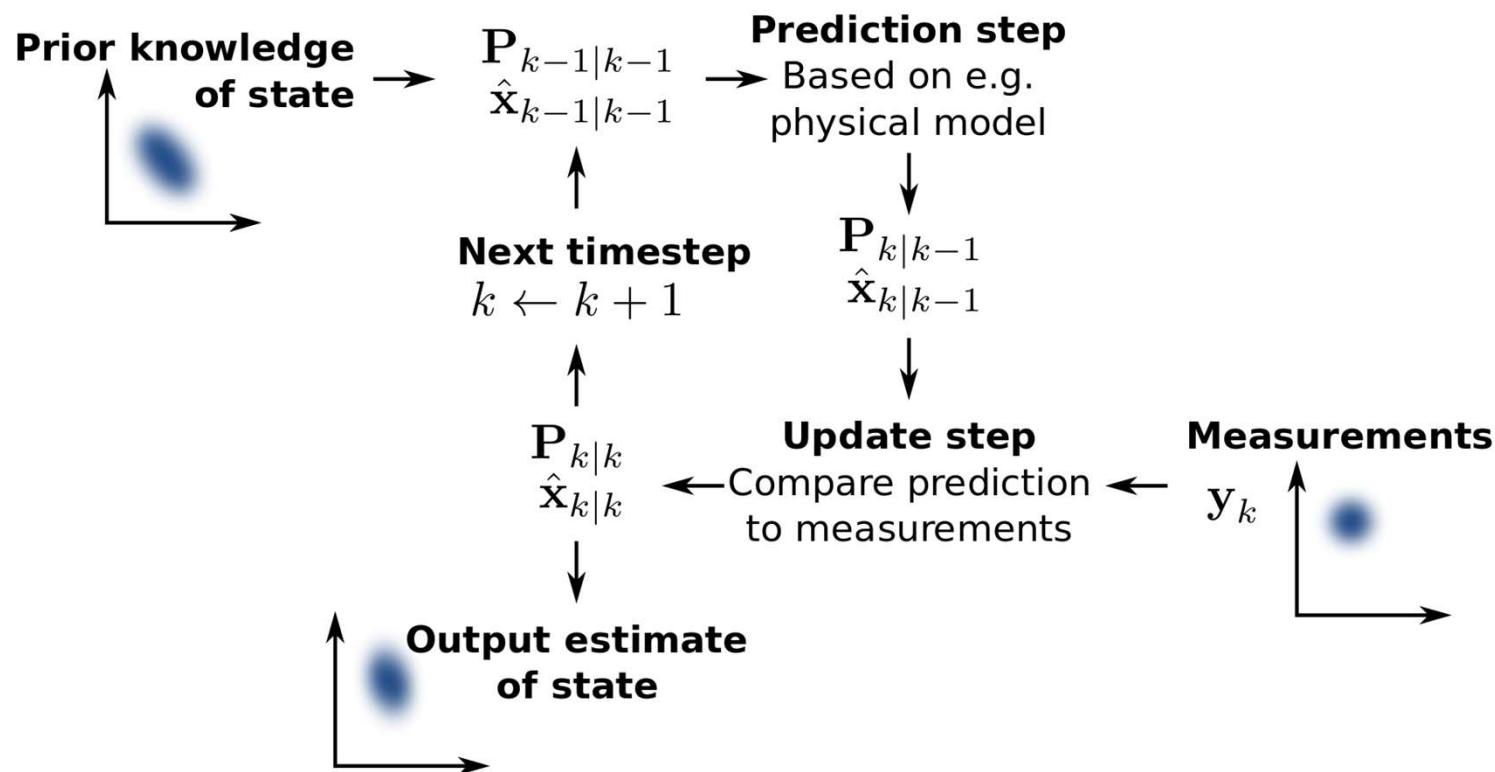
Kalman Filter

- Kalman-Filter (populärster Zustandsraum-Algorithmus): Satz von Rekursionen, zur Prädiktion/ Schätzung, Vorteil: iterative Bauart (PrädiktorKorrektor-Struktur) – ideal für Echtzeitanwendungen

Funktionsweise:

1. auf Basis des vorherigen Zustands $\hat{X}_{t-1|t-1}$: Vorhersage wahrscheinlichster Zustand $\hat{X}_{t|t-1}$ (Prädiktion)
2. Vergleich mit dem tatsächlich gemessenen Wert Y_t
3. Differenz der beiden Werte linear gewichten
4. Damit Verbesserung (Korrektur) des aktuellen Zustandes $\hat{X}_{t|t}$

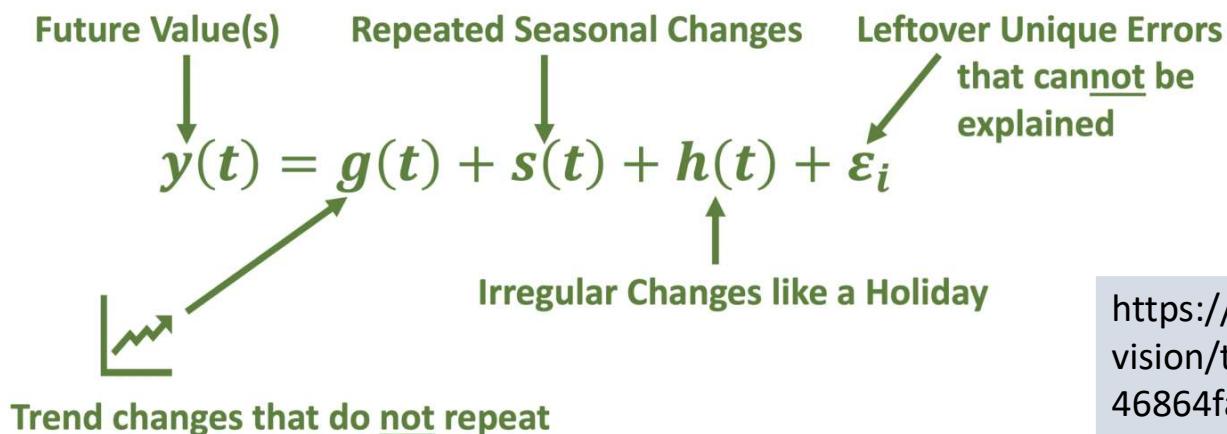
Kalman Filter



Additive Modelle

Facebook Prophet

- Prophet ist ein Verfahren zur Vorhersage von Zeitreihendaten auf der Grundlage eines additiven Modells, bei dem nichtlineare Trends mit jährlicher, wöchentlicher und täglicher Saisonalität sowie Ferieneffekten angepasst werden.



<https://medium.com/future-vision/the-math-of-prophet-46864fa9c55a>

Growth g(t)

Das stückweise lineare Modell wird anhand der folgenden statistischen Gleichungen angepasst,

$$y = \begin{cases} \beta_0 + \beta_1 x & x \leq c \\ \beta_0 - \beta_2 c + (\beta_1 + \beta_2) x & x > c \end{cases}$$

wobei c der Punkt der Trendänderung ist („Changepoint“).

Seasonality $s(t)$

- Die Saisonalitätsfunktion ist eine Fourier-Reihe in Abhängigkeit von der Zeit.

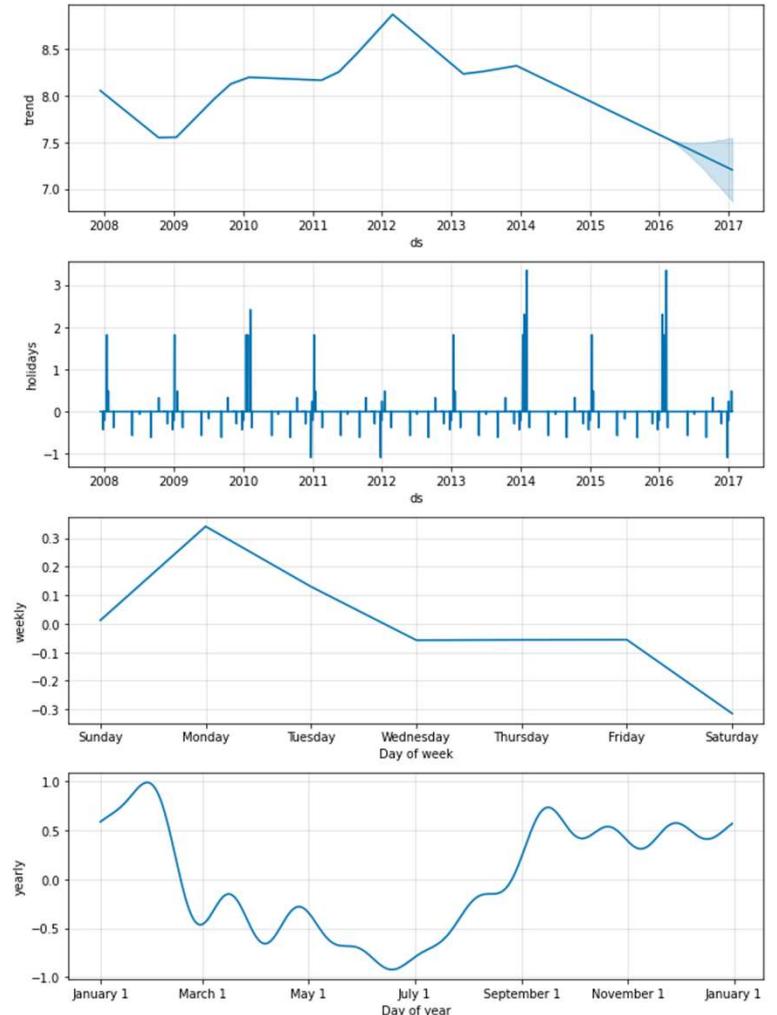
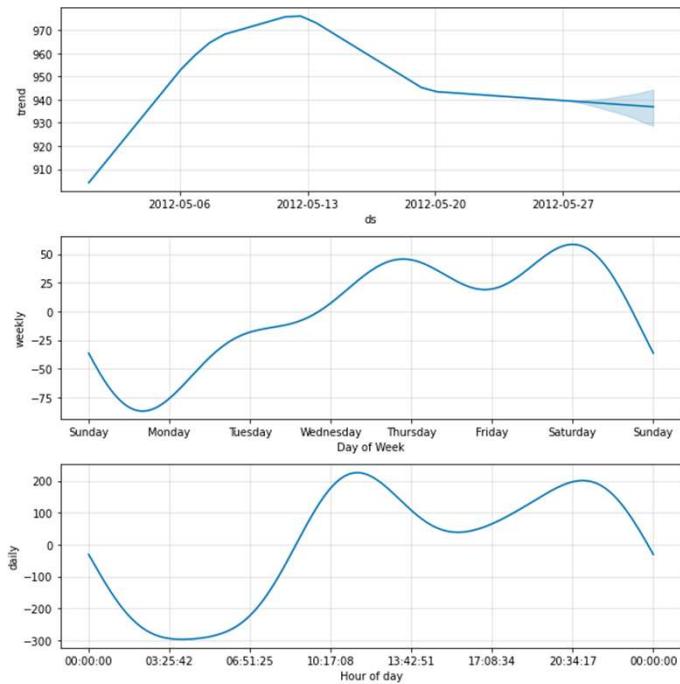
$$s(t) = \sum_{n=1}^N (a_n \cos(\frac{2\pi n t}{P}) + b_n \sin(\frac{2\pi n t}{P}))$$

Holidays $h(t)$

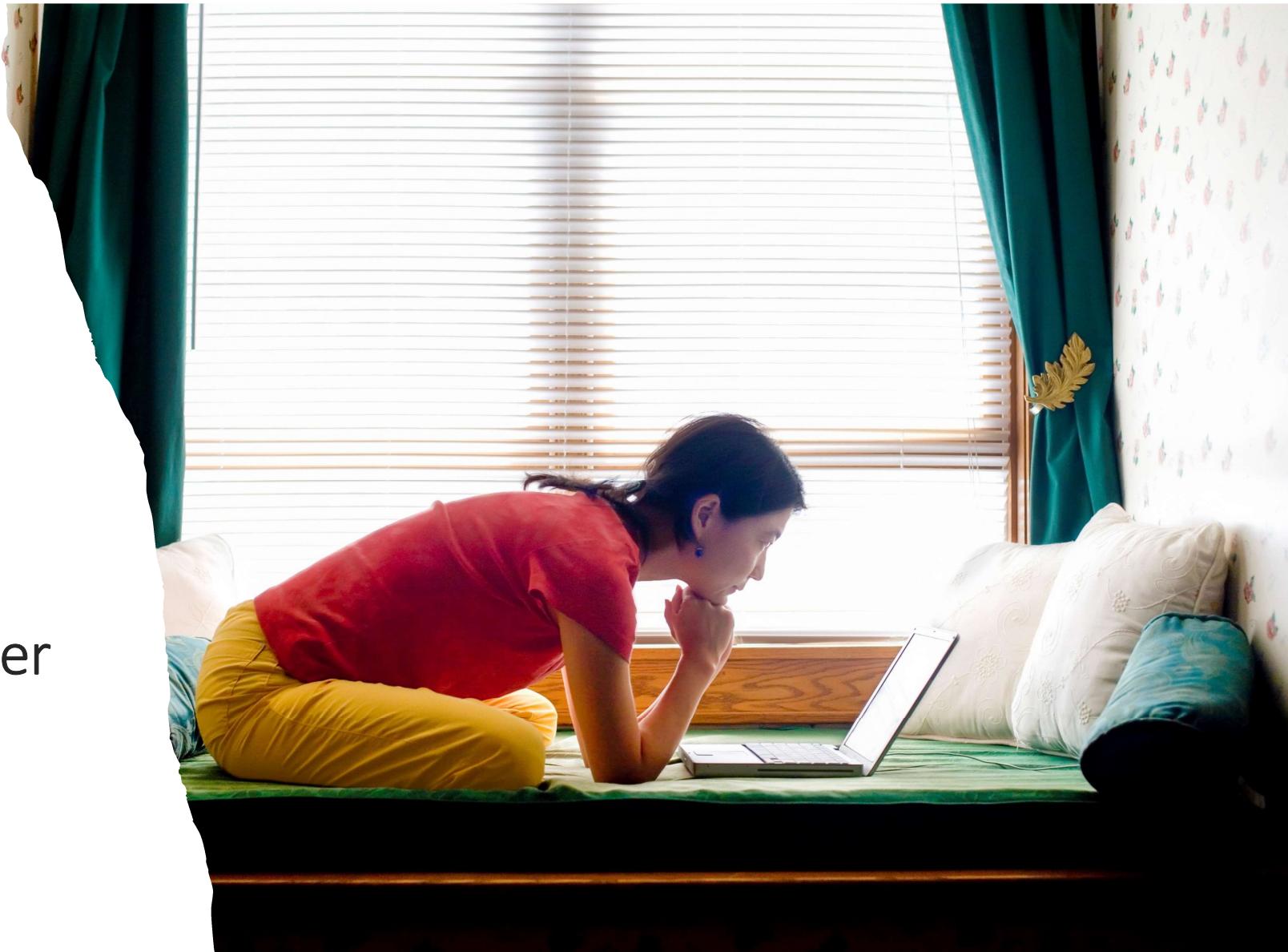
- Die Feiertagsfunktion ermöglicht es Facebook Prophet, die Vorhersage anzupassen, wenn ein Feiertag oder ein wichtiges Ereignis die Vorhersage verändern könnte.
- Sie nimmt eine Liste von Daten (es gibt eingebaute Daten von US-Feiertagen oder Sie können Ihre eigenen Daten definieren) und wenn jedes Datum in der Vorhersage vorhanden ist, addiert oder subtrahiert sie den Wert der Vorhersage von den Wachstums- und Saisonalitätsbedingungen, die auf historischen Daten zu den identifizierten Feiertagsdaten basieren.
- Man kann auch einen Bereich von Tagen um die Daten herum festlegen (z. B. die Zeit zwischen Weihnachten und Neujahr, Feiertagswochenenden, die Verbindung von Thanksgiving mit Black Friday/Cyber Monday usw.).

Zerlegung des additiven Modells

- Unterschiedliche Komponenten möglich



Ab ins Jupyter
Notebook



Machine Learning für Zeitreihen: Überblick, Klassifikation

Session 6 (Dienstag 11:00 – 12:30)



Machine Learning für Zeitreihen: Überblick, Vorbereitung und Vorhersagen

- Besonderheiten Machine Learning für Zeitreihen
- Random Forest Klassifikator, Bsp: Belegungserkennung
anhand von Raumklimadaten

KI, Machine Learning und Deep Learning für Zeitreihen

- Machine Learning und Deep Learning revolutionieren Umgang mit Daten
- Was ist mit Zeitreihen?
- Viele Verfahren sind nicht speziell für Zeitreihen (Ausnahmen später)
- Sie können aber für den Umgang mit Zeitreihen (sehr erfolgreich) genutzt werden
- Entscheidend ist:



KI, Machine Learning und Deep Learning für Zeitreihen

Artificial Intelligence (AI)



Component technologies

Expert systems

Robotics

Human interfaces
⋮

Machine learning



Algorithms

SVM

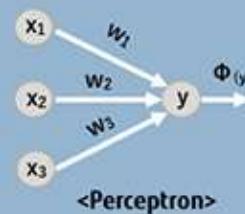
Random Forest

K-means
⋮

Reproduction of human intellectual activities

Learning of rules and patterns hidden in data

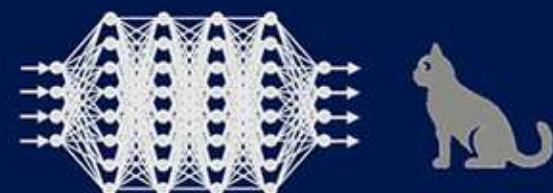
Neural networks



<Perceptron>
Modeling of neural signaling

Deep Learning

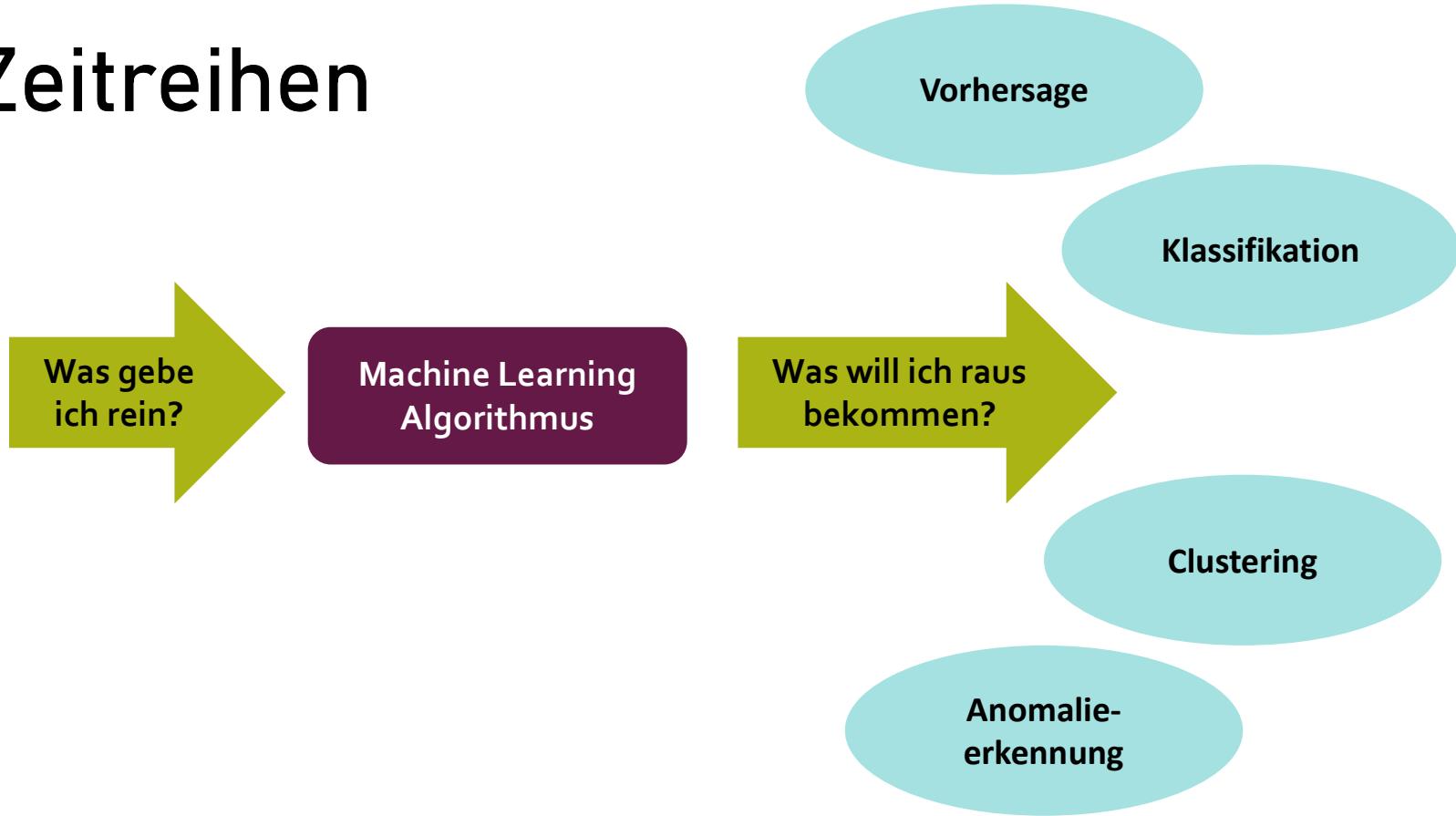
Learning using multi-layer neural networks

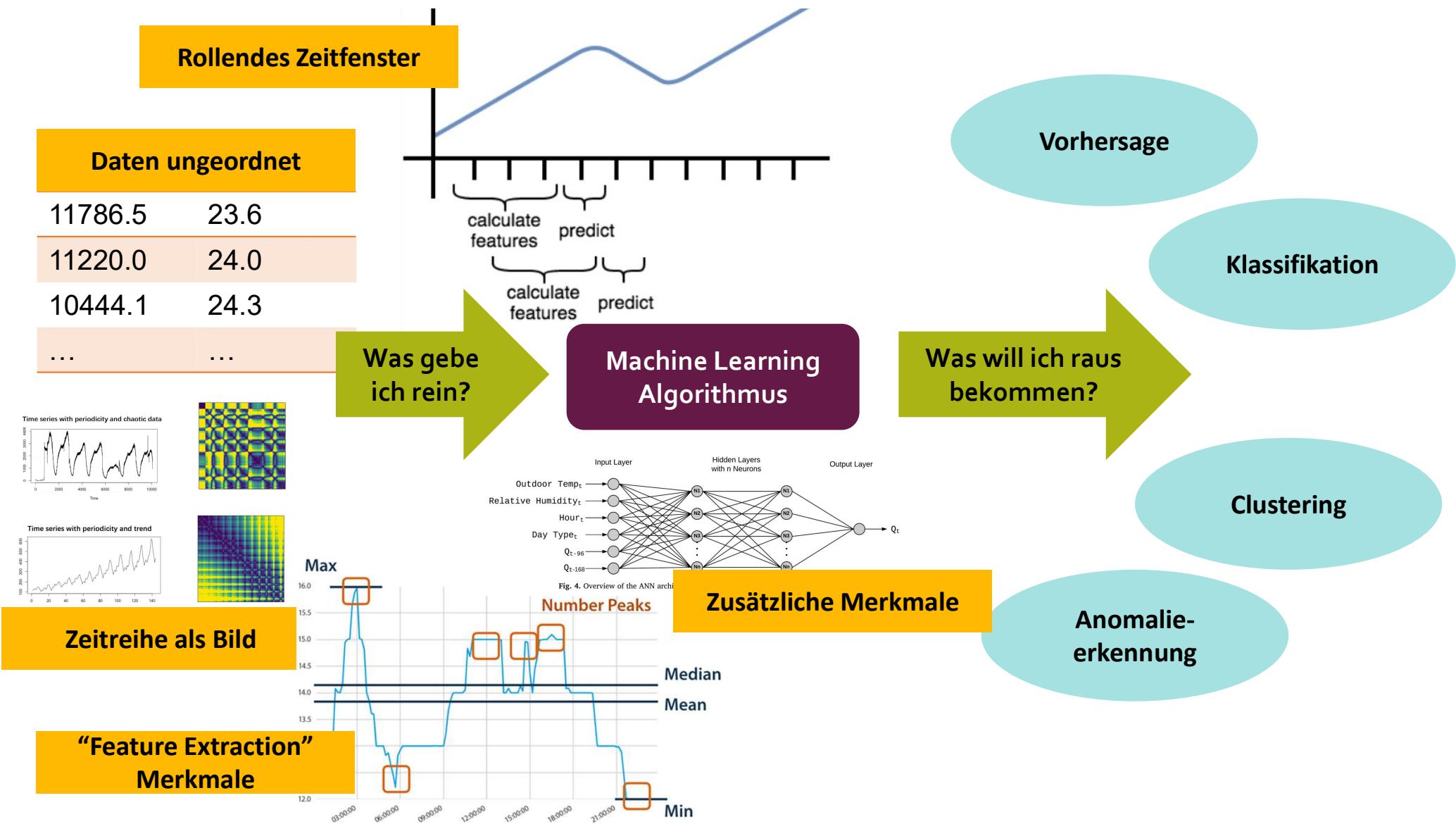


ML für Zeitreihen

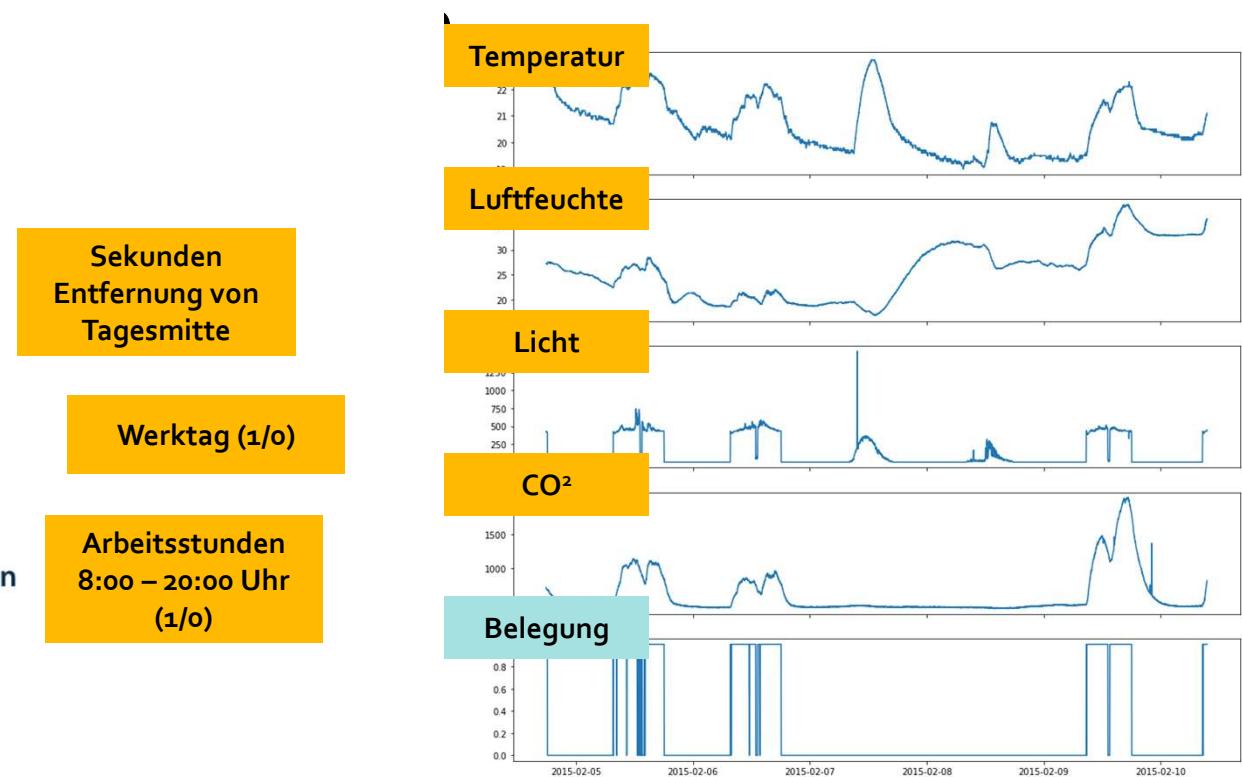
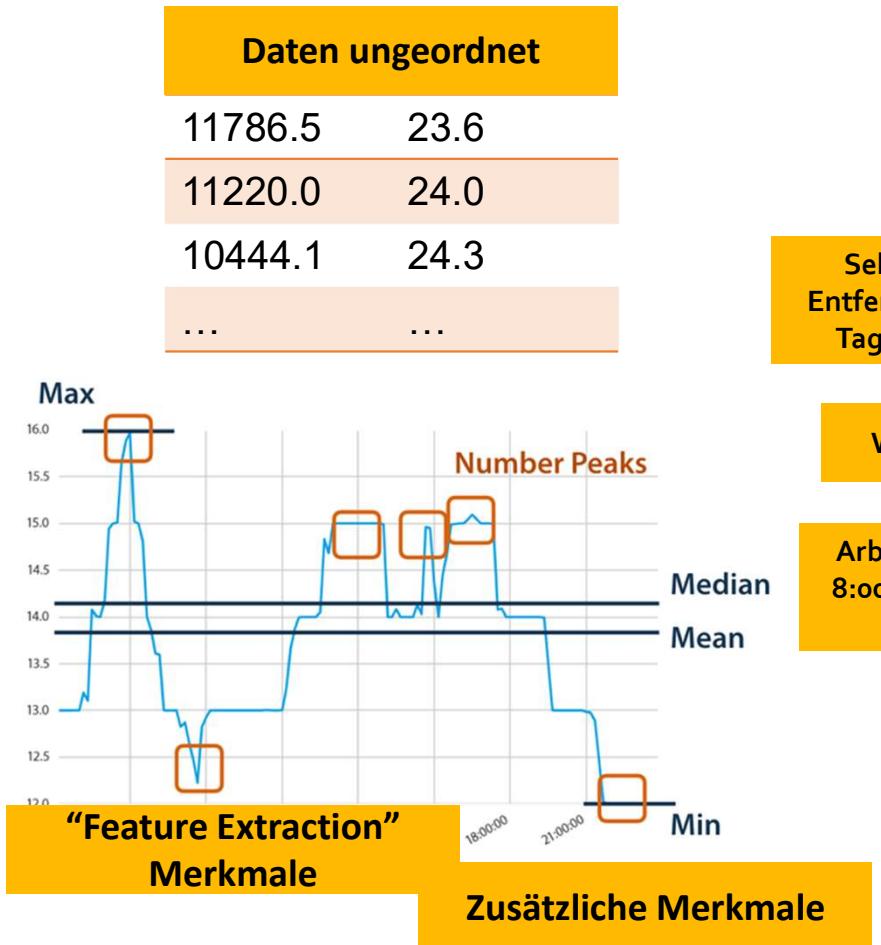


ML für Zeitreihen

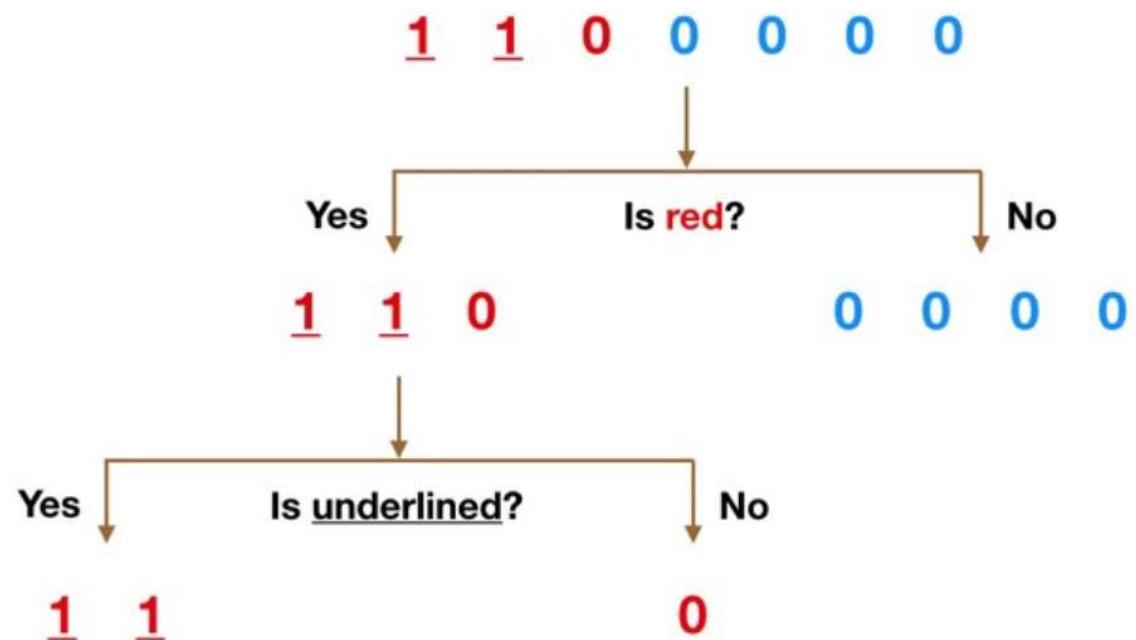




Beispiel: Belegungserkennung eines Büroraums aus Temperatur-, Feuchte-, Licht- und CO₂-Messungen



Entscheidungsbaum



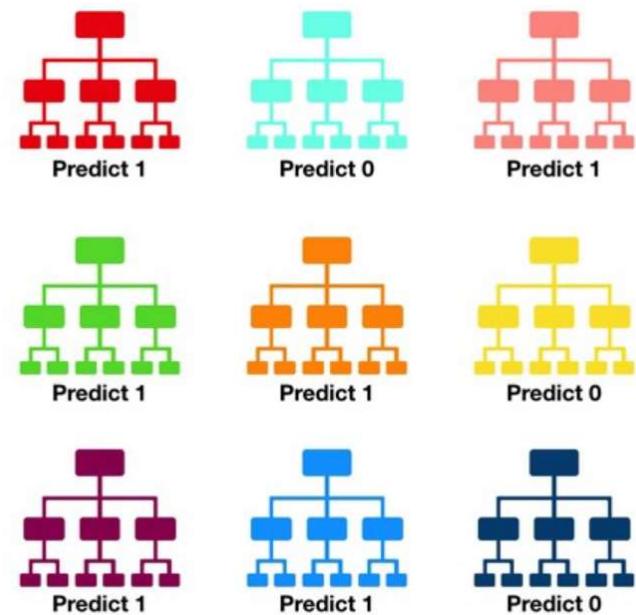
<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

Gini-Unreinheit (Gini Impurity)

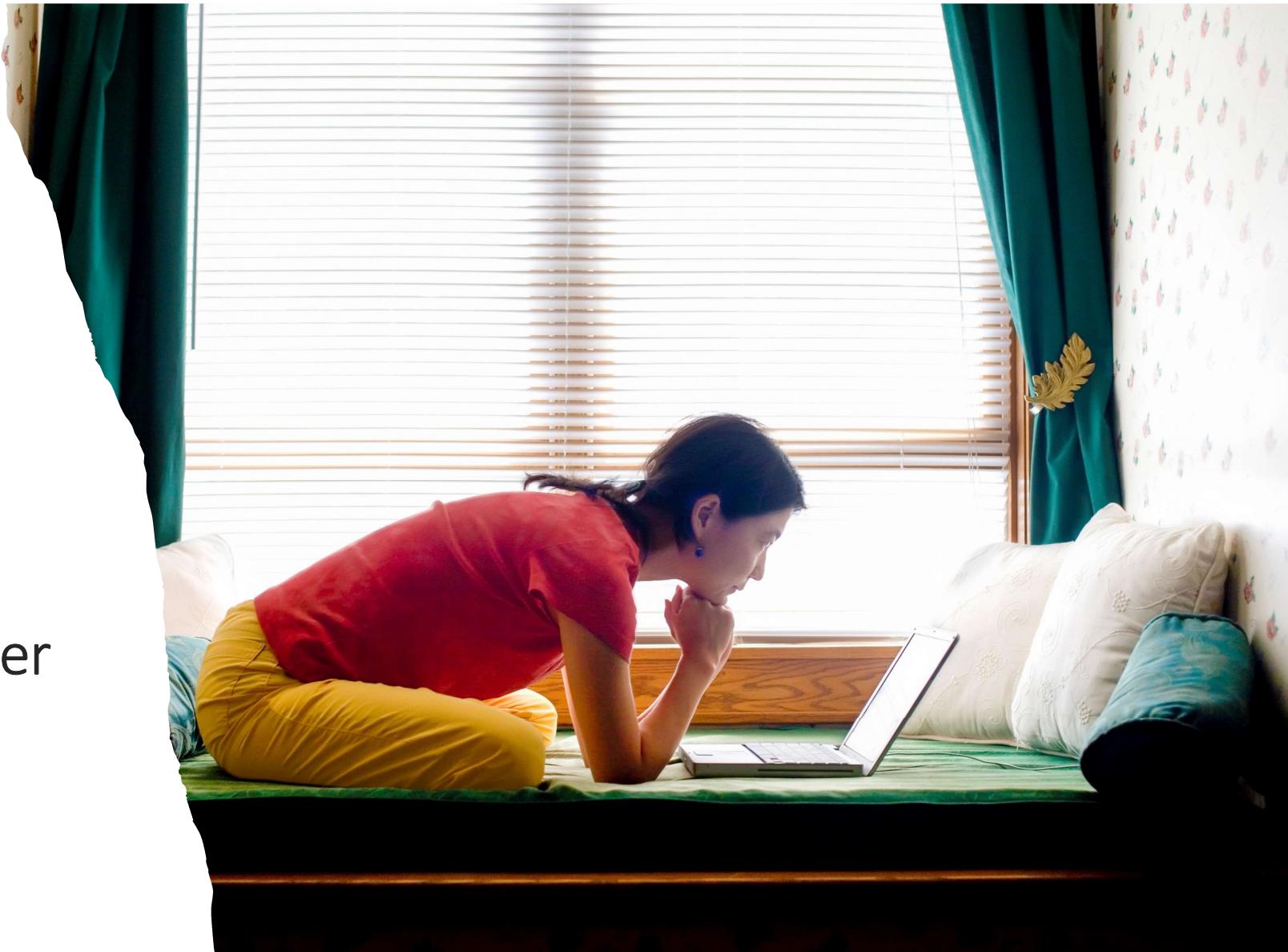
- **Gini-Unreinheit:** Maß dafür, wie oft ein zufällig ausgewähltes Element aus dem Datensatz falsch etikettiert würde, wenn es zufällig gemäß der Verteilung der Etiketten im Datensatz etikettiert würde.
- Die Gini-Unreinheit eines reinen Knotens, der nur Daten von einer einzigen Klasse enthält, ist 0.
- Entscheidungssplit: Für jeden möglichen Split berechnet der Entscheidungsbaum die Gini-Unreinheit für die Daten in den nachfolgenden Knoten.
- Der Split, der die größte Reduktion der Gini-Unreinheit bewirkt (oder alternativ den größten Information Gain liefert, wenn Entropie verwendet wird), wird als der beste Split ausgewählt.
- Rekursive Teilung: Prozess wird rekursiv fortgesetzt, bis entweder die Knoten eine bestimmte Mindestanzahl von Datenpunkten erreichen, der Baum eine vorher festgelegte Tiefe erreicht oder wenn keine weiteren Verbesserungen möglich sind.
- Praxis: Auch andere Stopp-Kriterien z.B. minimaler Verbesserungsschwellenwert für die Gini-Unreinheit

Random Forests

- Der Random Forest besteht, wie der Name vermuten lässt, aus einer großen Anzahl von einzelnen Entscheidungsbäumen, die als Ensemble arbeiten.
- Jeder einzelne Baum im Random Forest spuckt eine Klassen-vorhersage aus und die Klasse mit den meisten Stimmen wird zur Vorhersage unseres Modells



Ab ins Jupyter
Notebook



CODING

Belegungserkennung

- Klassifikation

Verbrauchergruppen

- Clustering



Machine Learning für Zeitreihen: Clustering

Session 7 (Dienstag 13:30 – 15:00)



Machine Learning für Zeitreihen: Clustering und Klassifikation

- Clustering, Beispiel: Städteklimacluster
- Clustering, Beispiel: Verbrauchercluster

Zeitreihen Clustering

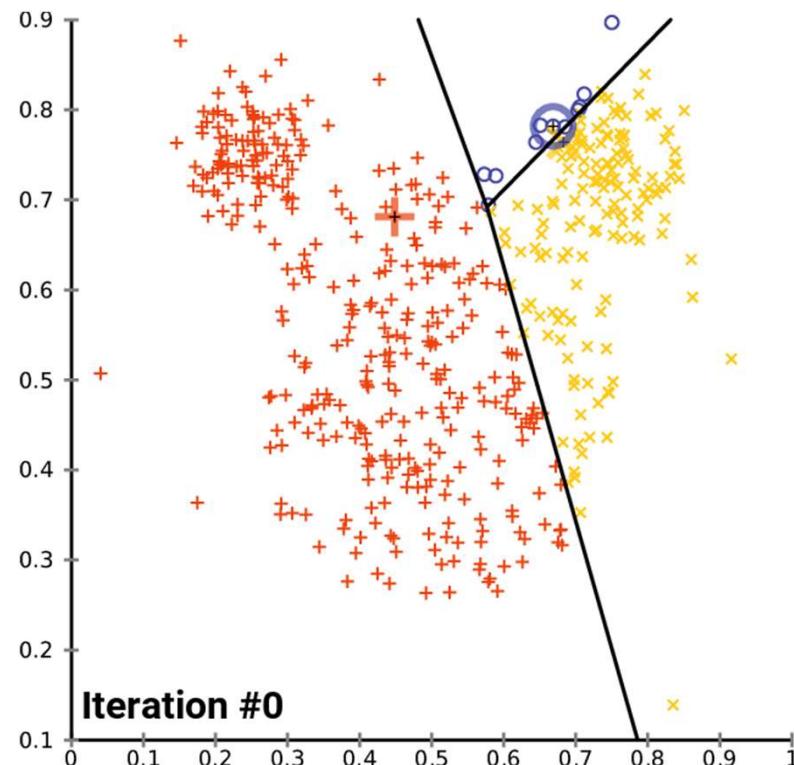
- Zeitreihen Clustering = Gruppierung von Zeitreihendaten basierend auf Ähnlichkeiten
- Hilft Muster zu identifizieren
- Praxis aber auch Gruppierung selbst als Ziel
- Verschiedene Clustering-Methoden:
 - k-Means-Clustering
 - Hierarchisches Clustering

k-Means-Clustering

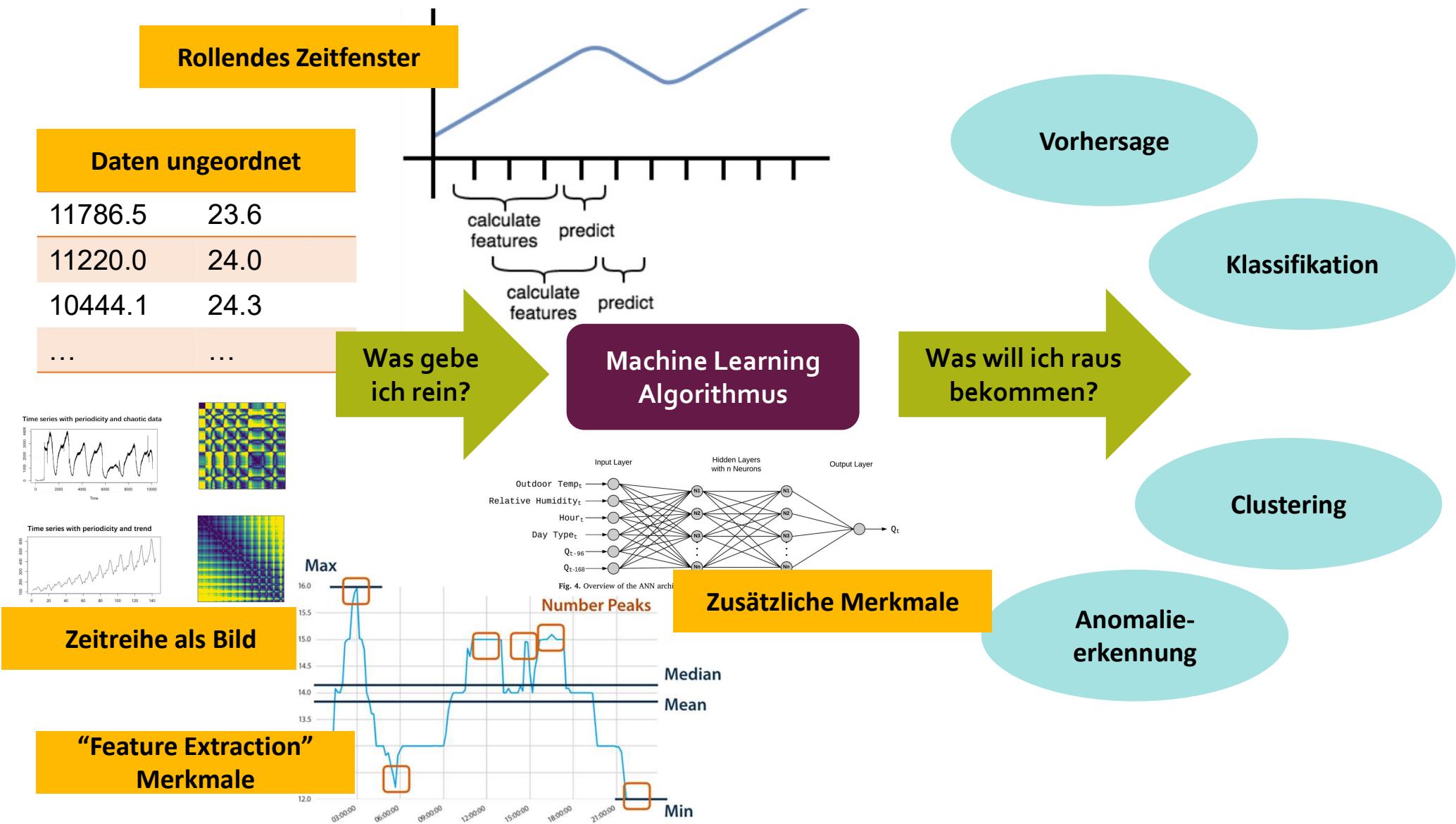
Grundprinzip:

- k-Means ist ein partitionierendes Clustering-Verfahren, das Datenpunkte in k Gruppen (Cluster) aufteilt, basierend auf deren Merkmalen.
- Das Ziel ist es, die Summe der Quadrate der Distanzen zwischen den Datenpunkten und dem jeweiligen Clusterzentrum zu minimieren.
- Anwendung auf Zeitreihen: Bei der Anwendung auf Zeitreihen werden die Zeitreihendaten so gruppiert, dass Zeitreihen im selben Cluster ähnliche Muster (wie Trends oder zyklische Bewegungen) aufweisen.
- Dies erfordert oft die Anpassung oder Transformation der Zeitreihendaten, damit sie mit der k-Means-Methodik kompatibel sind.

K-Means Clustering



<https://towardsdatascience.com/hands-on-climate-time-series-clustering-using-machine-learning-with-python-6a12ce1607f9>



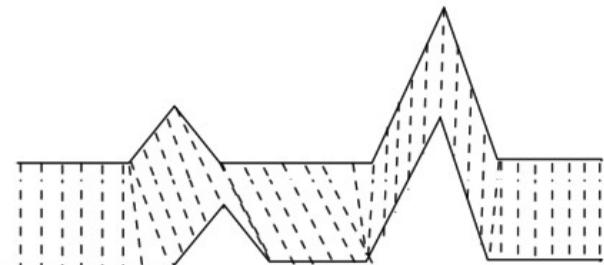
Hierarchisches Clustering

Grundprinzip:

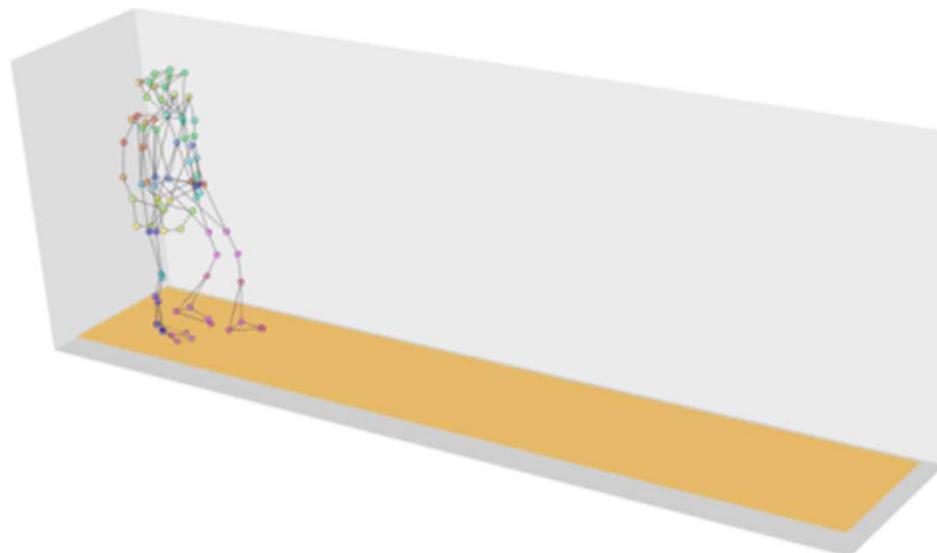
- Im Gegensatz zu k-Means, das eine feste Anzahl von Clustern erfordert, bildet das hierarchische Clustering eine Baumstruktur (Dendrogramm), die zeigt, wie jeder Datenpunkt in kleinere Gruppen aufgeteilt wird.
- Zwei Ansätze: agglomerativ (von unten nach oben, beginnend mit jedem Datenpunkt als eigenem Cluster) und divisiv (von oben nach unten)
- Visualisierung und Funktion: Jeder Schritt des agglomerativen Prozesses kann visualisiert werden, wobei Datenpunkte oder bestehende Cluster, die sich am ähnlichsten sind, schrittweise zusammengeführt werden. Das Ergebnis ist ein Dendrogramm, das die Hierarchie und Ähnlichkeitsbeziehungen darstellt

Clustering für Zeitreihen

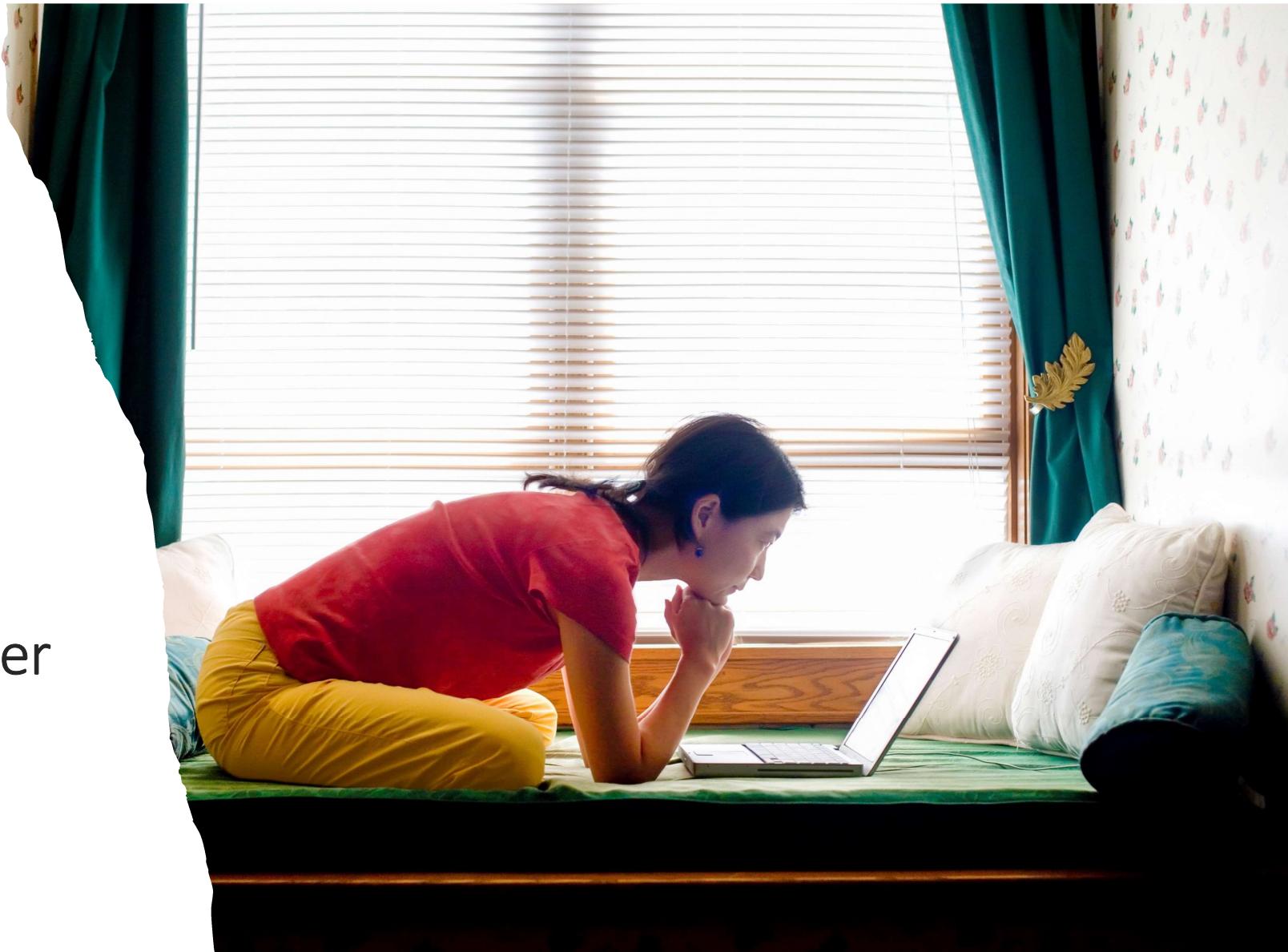
- Warum brauchen wir ein eigenes Verfahren?
- Problem: Wie messe ich den „Abstand“ zwischen zwei Zeitreihen?
- mit traditionellen Methoden (MAE, MSE) berechnete Abstand immer noch sehr groß, selbst wenn der Unterschied zwischen den beiden Zeitreihen im Grunde 0 ist (z.B. leicht verschoben)
- Stattdessen: Distance Time Warping



Distance Time Warping



Ab ins Jupyter
Notebook



CODING

Belegungserkennung

- Klassifikation



Deep Learning für Zeitreihen

Session 8 (Dienstag 15:15 – 17:00)

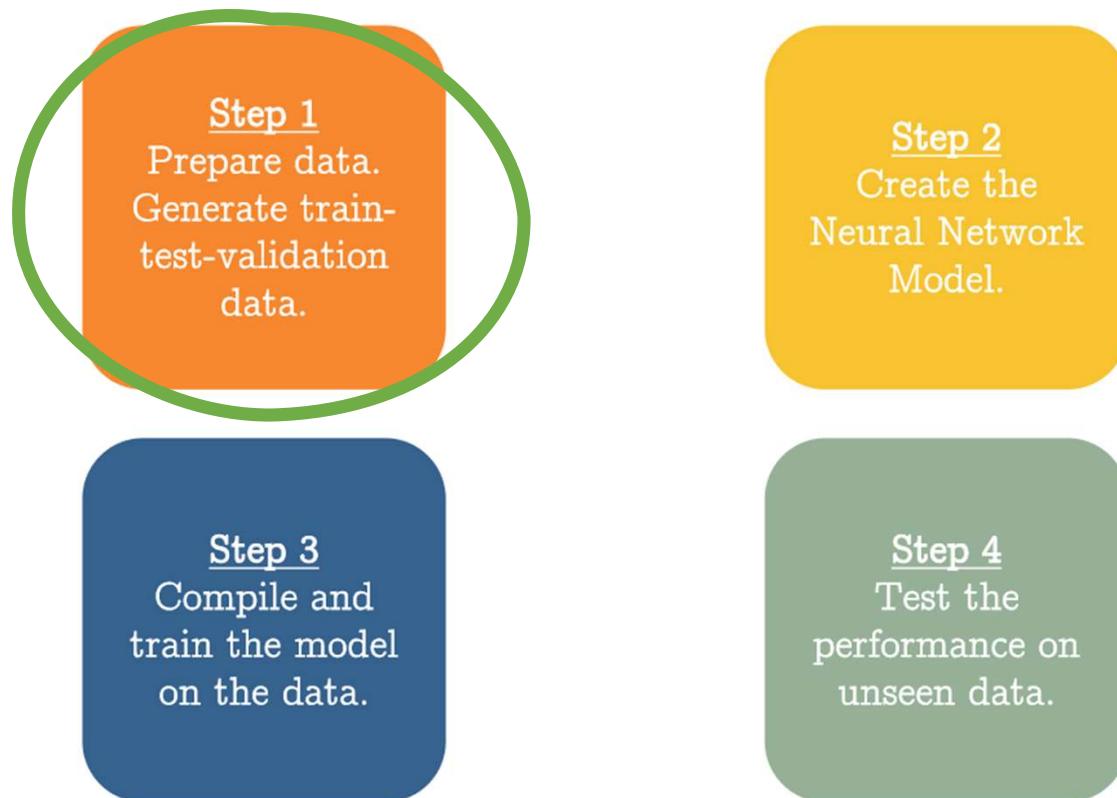


Deep Learning für Zeitreihenvorhersage

- Datenvorbereitung
- Recurrent Neural Networks (RNN)
- LSTMs

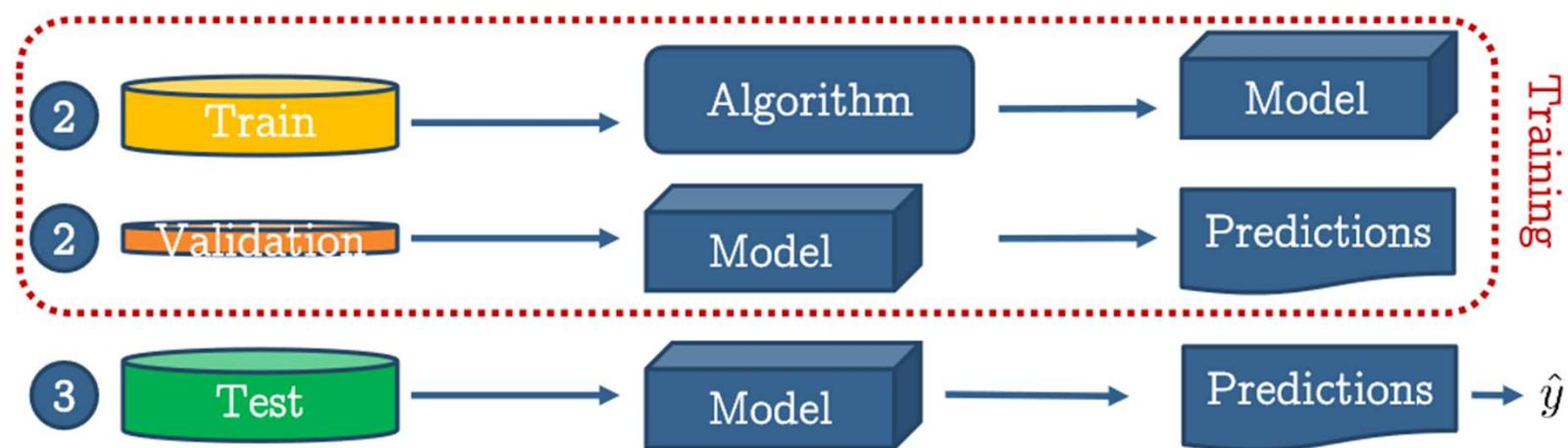
Datenvorbereitung: Trainings- und Testdaten

Datenvorbereitung für Deep Learning Zeitreihenanalyse



Datenvorbereitung: Trainings- und Testdaten

Datenvorbereitung für Deep Learning Zeitreihenanalyse



Datenvorbereitung: Trainings- und Testdaten

Datenvorbereitung für Deep Learning Zeitreihenanalyse

- Zeitreihendaten sind einzigartig: **zeitliche Reihenfolge der Beobachtungen spielt eine (große) Rolle**
- **Datenvorbereitung** für Aufgaben des maschinellen Lernens und des Deep Learning ist etwas komplexer
- Übliche **zufällige Aufteilung** von Trainings- und Testdaten oder das **Mischen von Daten ist nicht geeignet**, da dadurch die zeitliche Anordnung der Daten gestört wird
- Hier sind einige gängige Ansätze zur Vorbereitung von Zeitreihendaten (für Deep Learning):
 - Sequentieller Split
 - Stapelung von Sequenzen
 - Rolling Window
 - Expanding Window
- Auch zwischen Epochs, müssen wir aufpassen, denn zufälliges Mischen kommt für Zeitreihen nicht in Frage.

Sequentieller Split: Training/Validierung/Test

Datenvorbereitung für Deep Learning Zeitreihenanalyse

- Dies ist der einfachste und am häufigsten verwendete Ansatz für die Aufteilung von Zeitreihendaten.
- Sie ordnen die Daten nach den Zeitstempeln und unterteilen sie dann in drei Segmente: Trainings-, Validierungs- und Testsätze.
- Wenn wir beispielsweise 10 Monate an Daten haben, können wir die ersten 6 Monate für das Training, die nächsten 2 Monate für die Validierung und die letzten 2 Monate für den Test verwenden.
- Dieser Ansatz respektiert die zeitliche Anordnung der Daten und ist einfach zu implementieren.



Stapelung von Sequenzen

Datenvorbereitung für Deep Learning Zeitreihenanalyse

- Ein anderer Ansatz besteht darin, die Zeitreihendaten in sich nicht überschneidende Sequenzen zu unterteilen.
- Jede Sequenz behält ihre ursprüngliche Reihenfolge bei, aber die Sequenzen können zwischen den Epochs gemischt werden.
- Sequentieller Split:



- Stapelung von Sequenzen:



Rollendes und Expandierendes Fenster

Datenvorbereitung für Deep Learning Zeitreihenanalyse

- Hier wächst das Trainingfester an und verschiebt sich nicht wie beim „Rolling Window“
- Rolling Window:

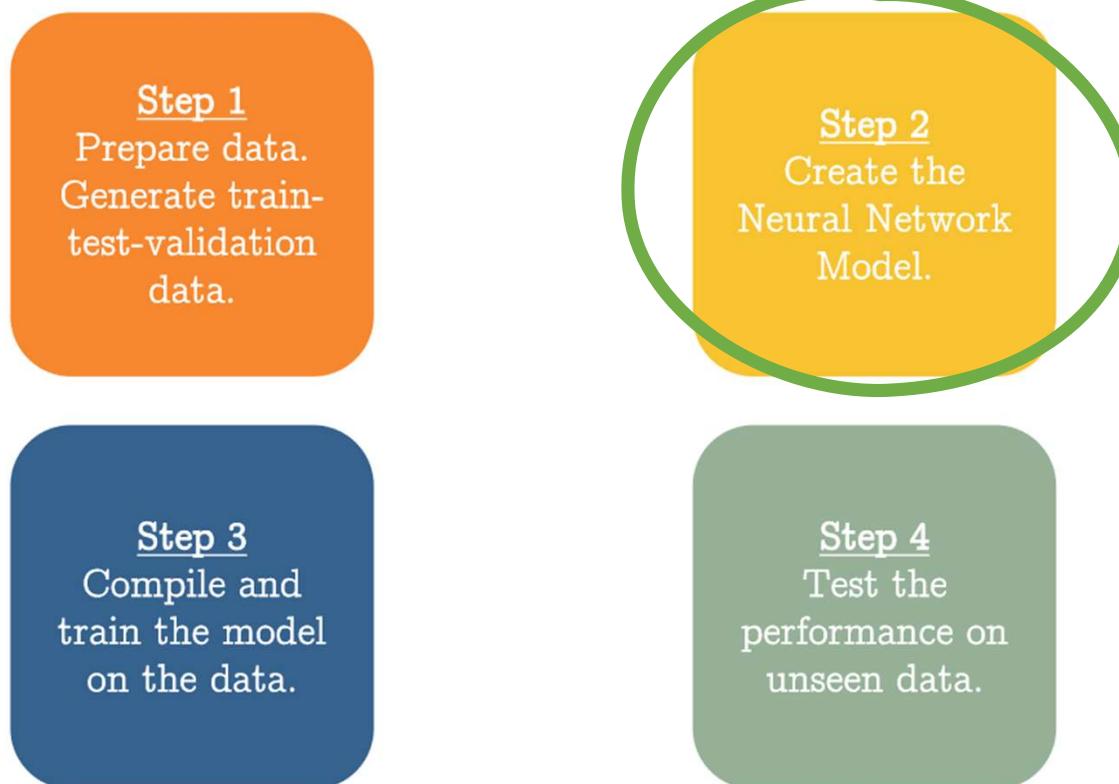


Expanding Window:



Datenvorbereitung: Trainings- und Testdaten

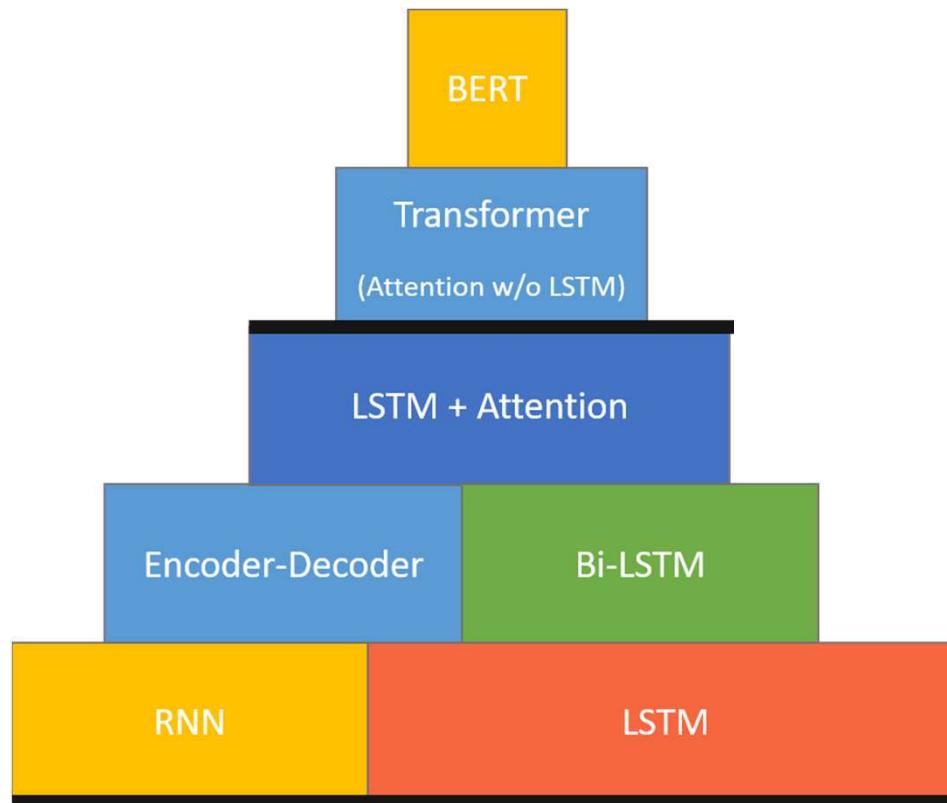
Datenvorbereitung für Deep Learning Zeitreihenanalyse



RNNs als Grundlage für modernste Verfahren

Rekurrente Neuronale Netze (RNNs)

- „BERT Mountain“

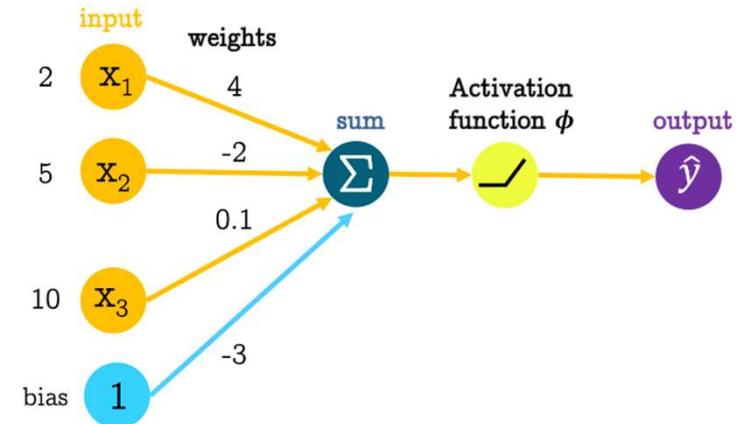


Quelle: <https://mccormickml.com/2019/11/11/bert-research-ep-1-key-concepts-and-sources/>

Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

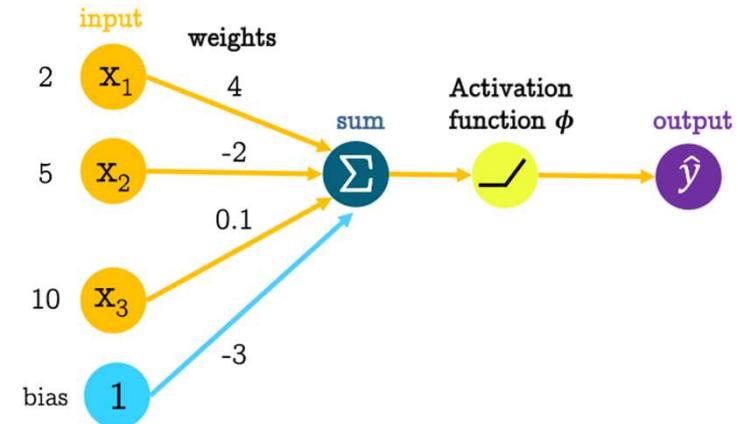
- Warum nehmen wir nicht einfach ein Perzeptron für Zeitreihendaten?
- **Zeitliche Abhängigkeiten:**
 - Zeitreihendaten weisen oft zeitliche Abhängigkeiten auf, was bedeutet, dass der aktuelle Wert von früheren Werten abhängt.
 - Ein normales Perzeptron berücksichtigt die zeitliche Reihenfolge der Daten nicht, da es jede Eingabe unabhängig behandelt.
 - Es ist nicht in der Lage, diese Abhängigkeiten zu erfassen und zu modellieren, die für genaue Vorhersagen in der Zeitreihenanalyse entscheidend sind.



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

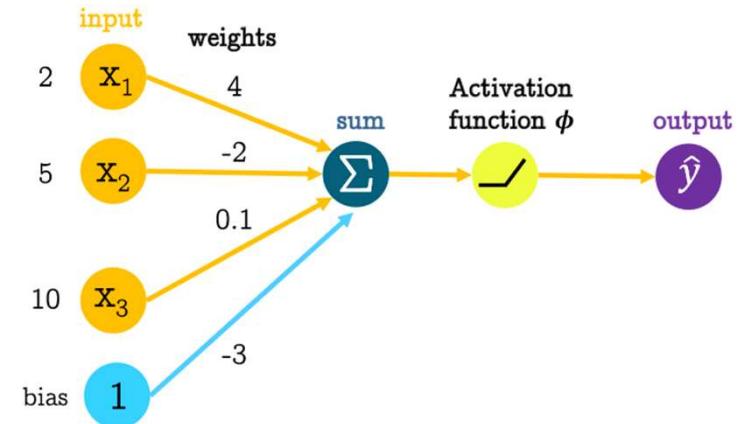
- Warum nehmen wir nicht einfach ein Perzeptron für Zeitreihendaten?
- **Eingaben mit variabler Länge:**
 - Zeitreihendaten können variable Längen haben, da die Anzahl der Beobachtungen variieren kann.
 - Normale Perzeptrons erfordern in der Regel Eingabevektoren fester Länge, so dass die direkte Verarbeitung von Eingaben variabler Länge nicht einfach ist.



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

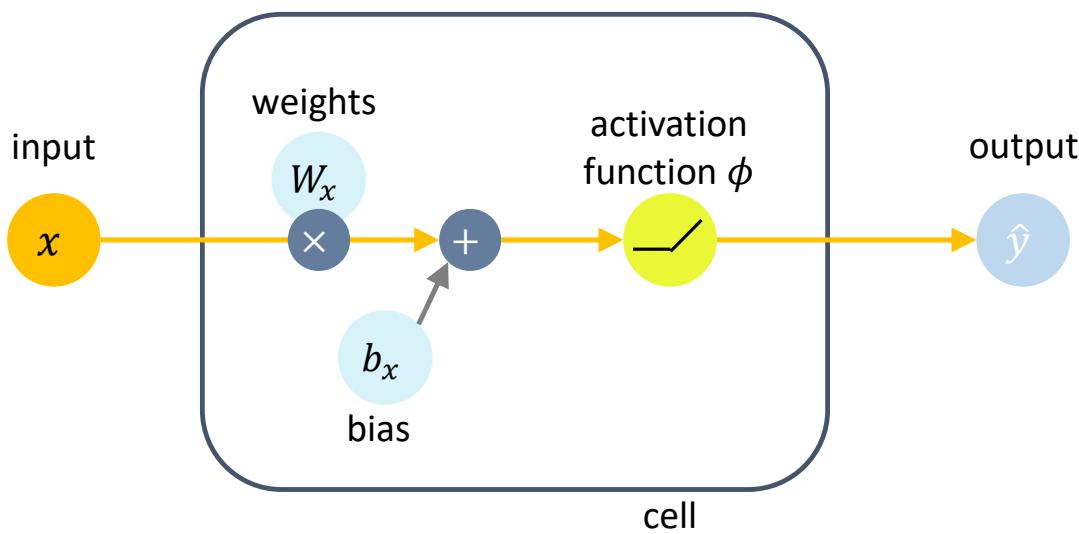
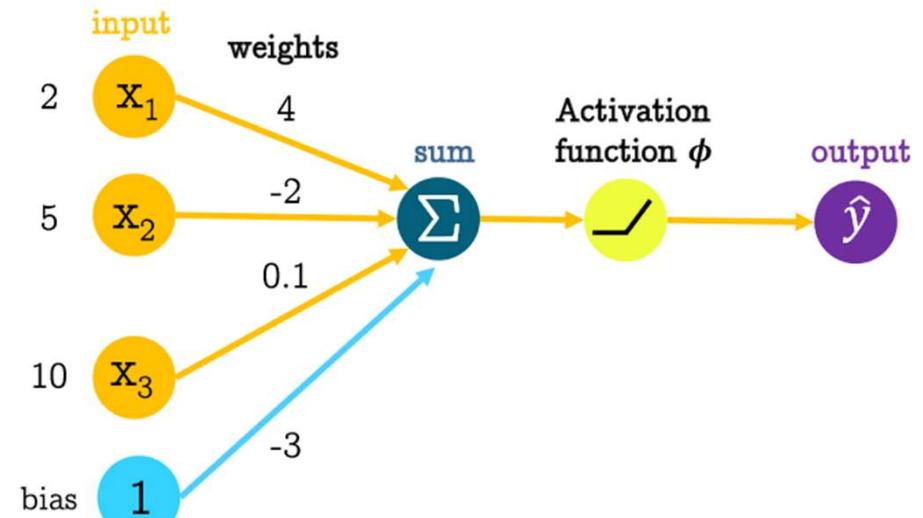
- Warum nehmen wir nicht einfach ein Perzeptron für Zeitreihendaten?
- **Komplexe Muster:**
 - Zeitreihendaten können komplexe Muster wie Trends, Saisonalität und langfristige Abhängigkeiten aufweisen.
 - Diese Muster erfordern unter Umständen eine ausgereiftere Modellarchitektur, um die zeitliche Dynamik effektiv zu erfassen.



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

- Perzepron – eine andere Darstellung
- Nur ein Input (kann ein Vektor / Tensor sein)
- Gewichtsmatrix W

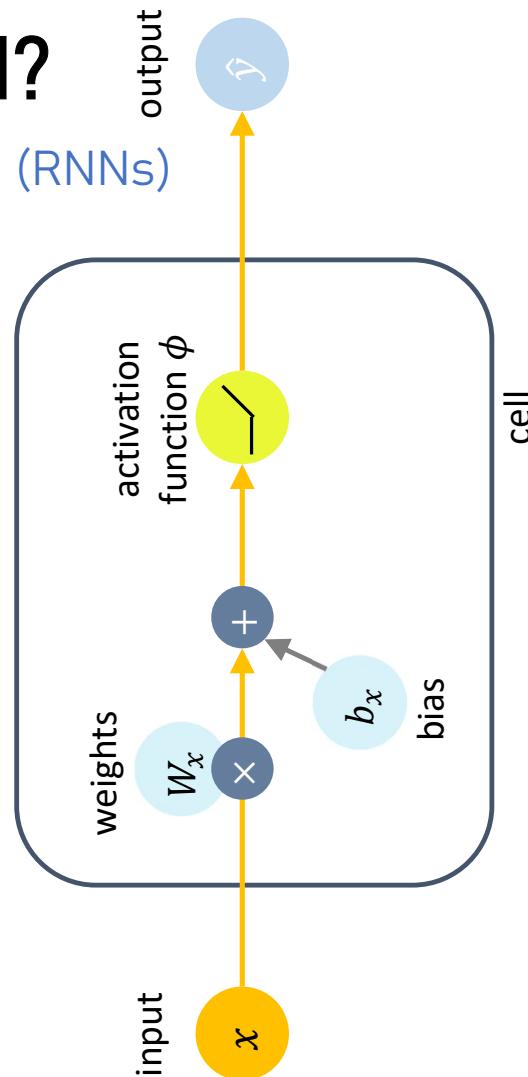


$$\hat{y} = \phi(W_x x + b_x)$$

Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

- Perzeptron

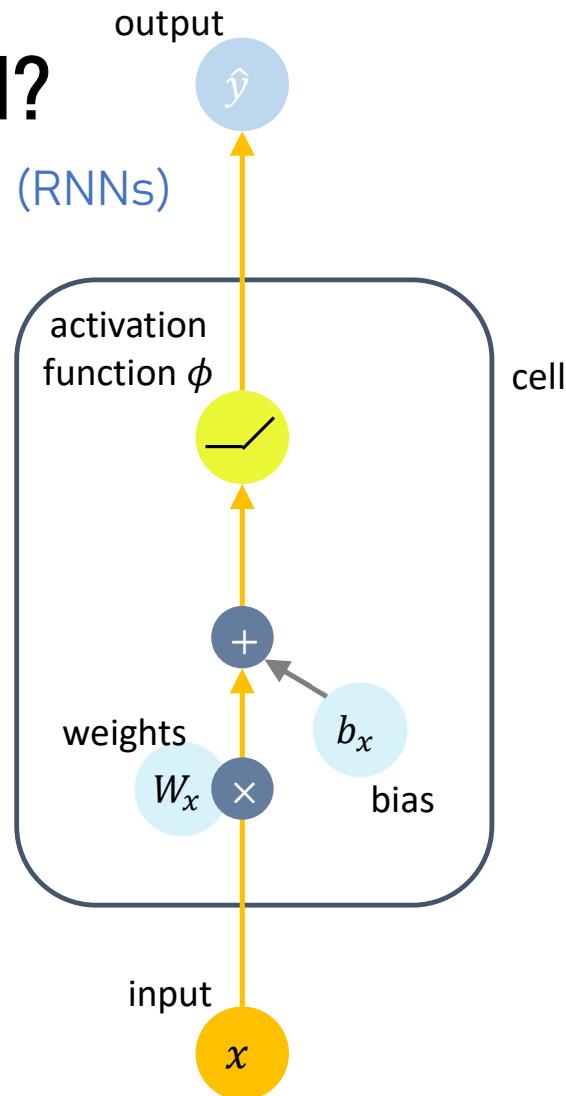


$$\hat{y} = \phi(W_x x + b_x)$$

Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

- Perzeptron

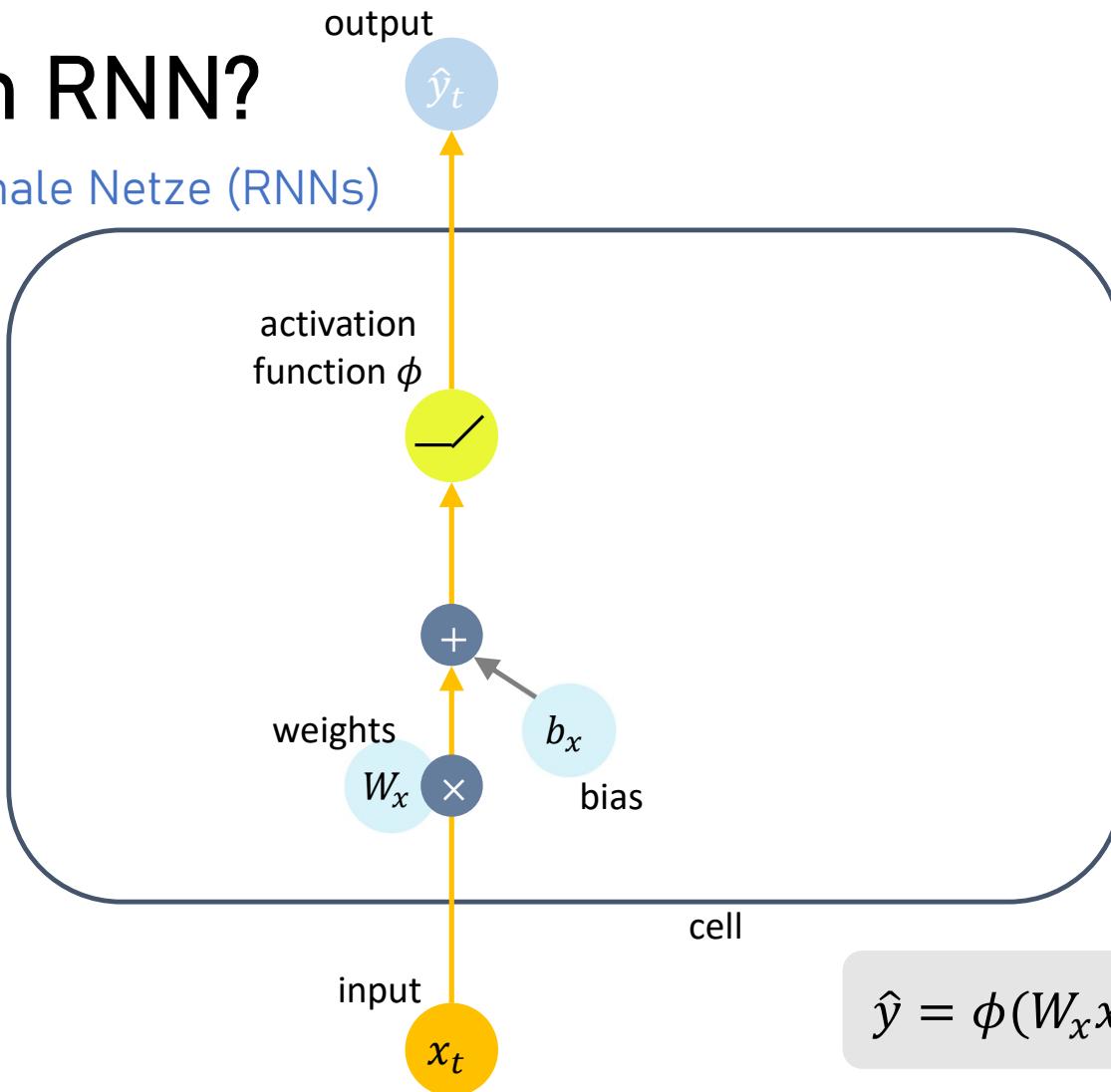


$$\hat{y} = \phi(W_x x + b_x)$$

Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

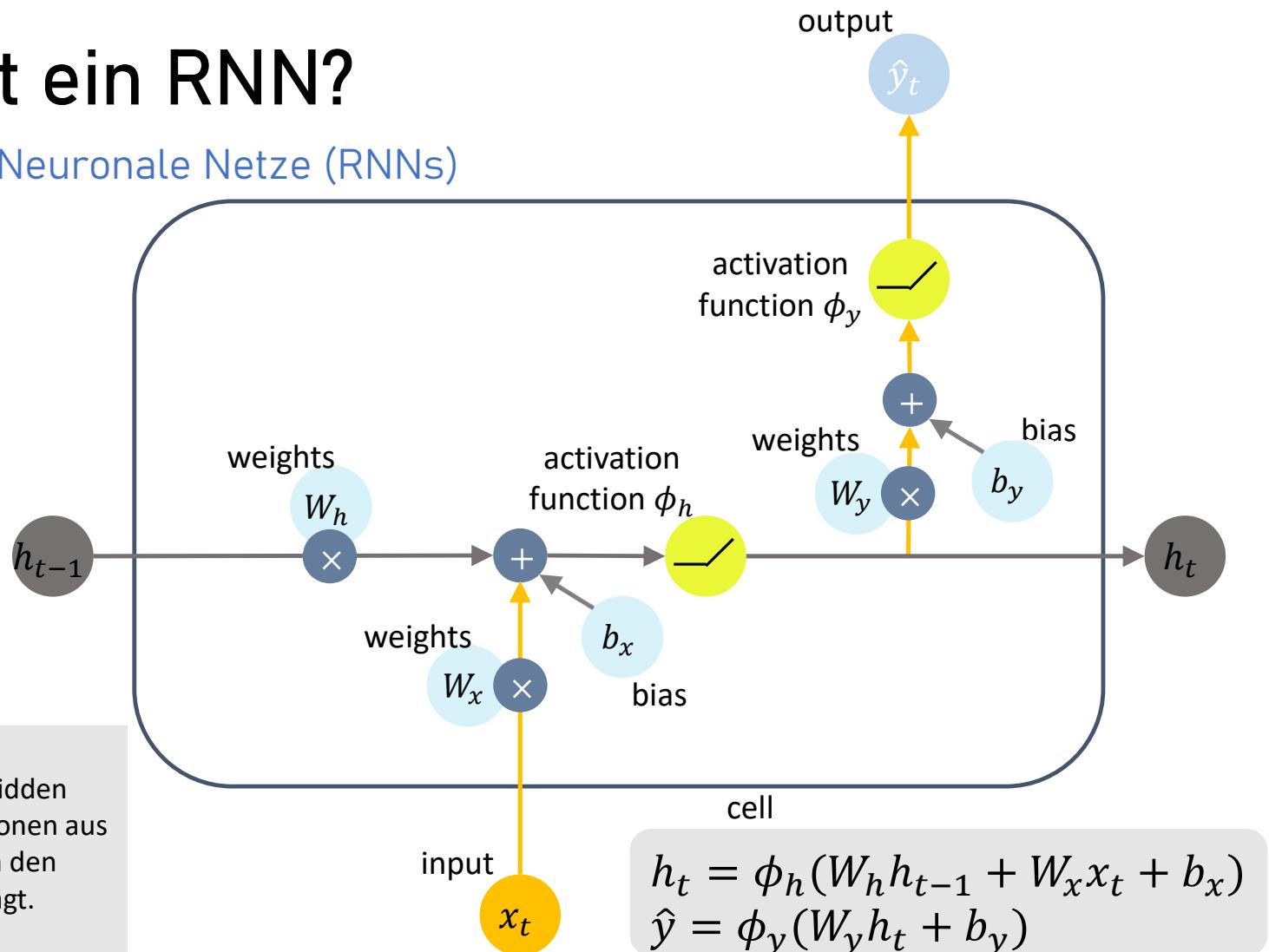
- Perzeptron



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

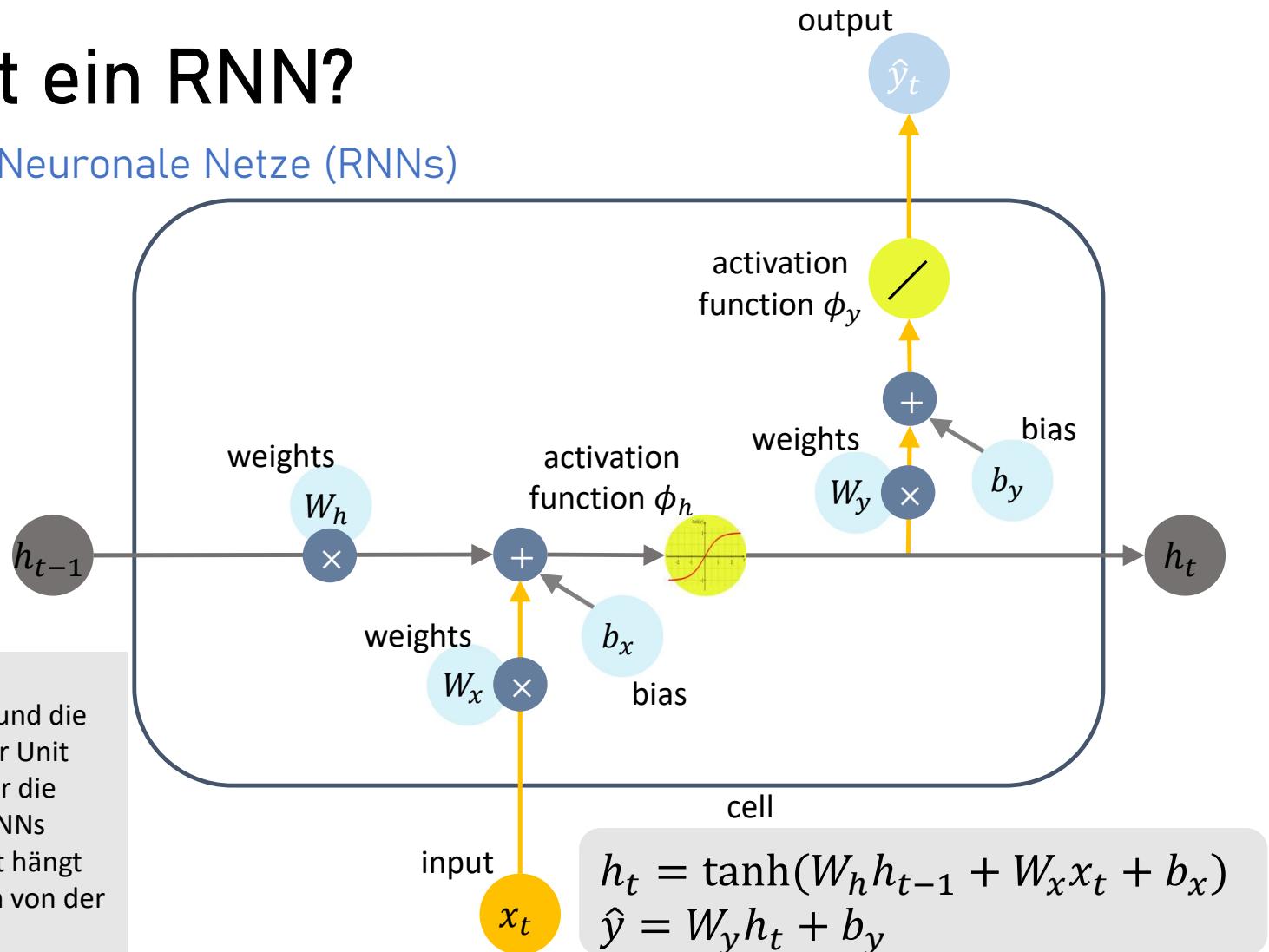
- RNN Zelle



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

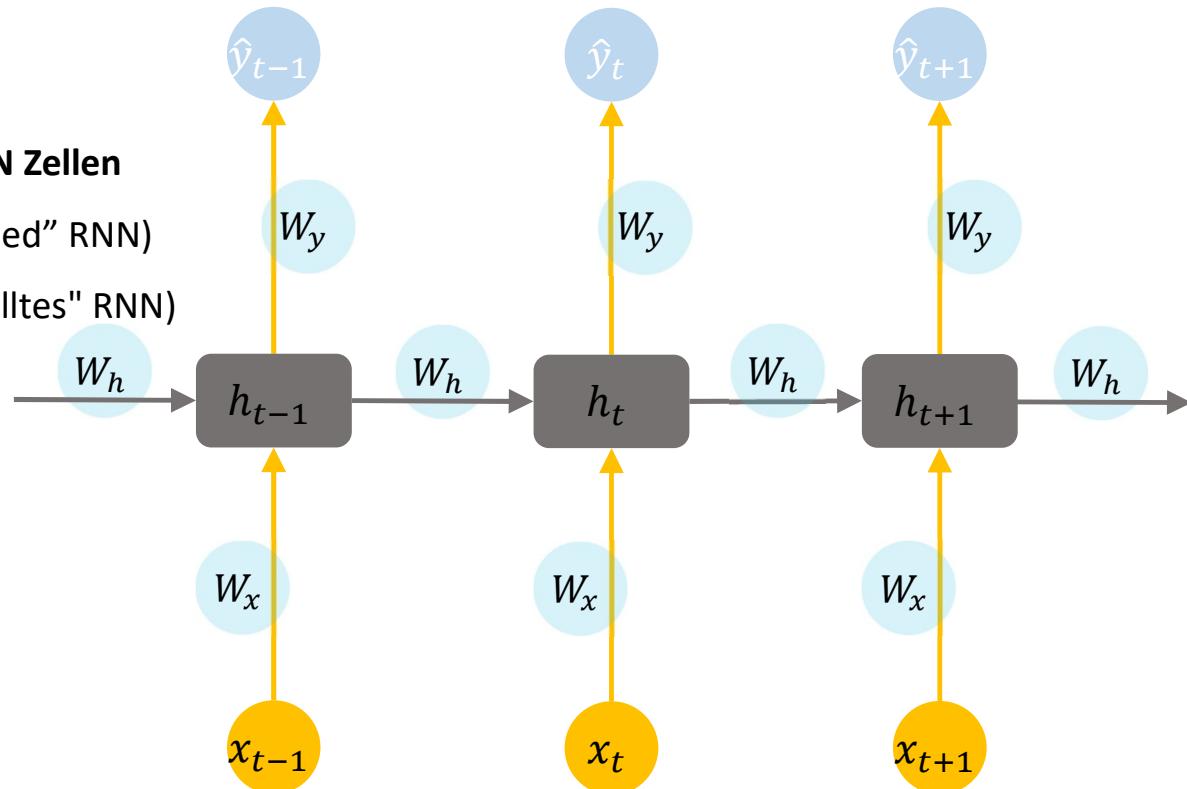
- RNN Zelle



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

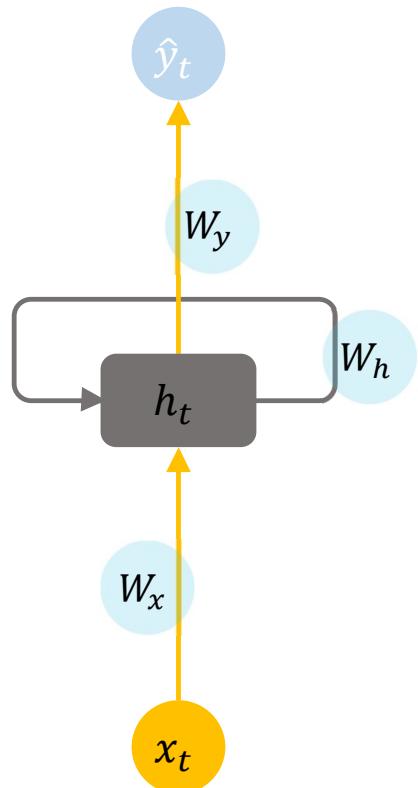
- **Mehrere RNN Zellen**
- (engl. "unrolled" RNN)
- (dt. "ausgerolltes" RNN)



Was ist ein RNN?

Rekurrente Neuronale Netze (RNNs)

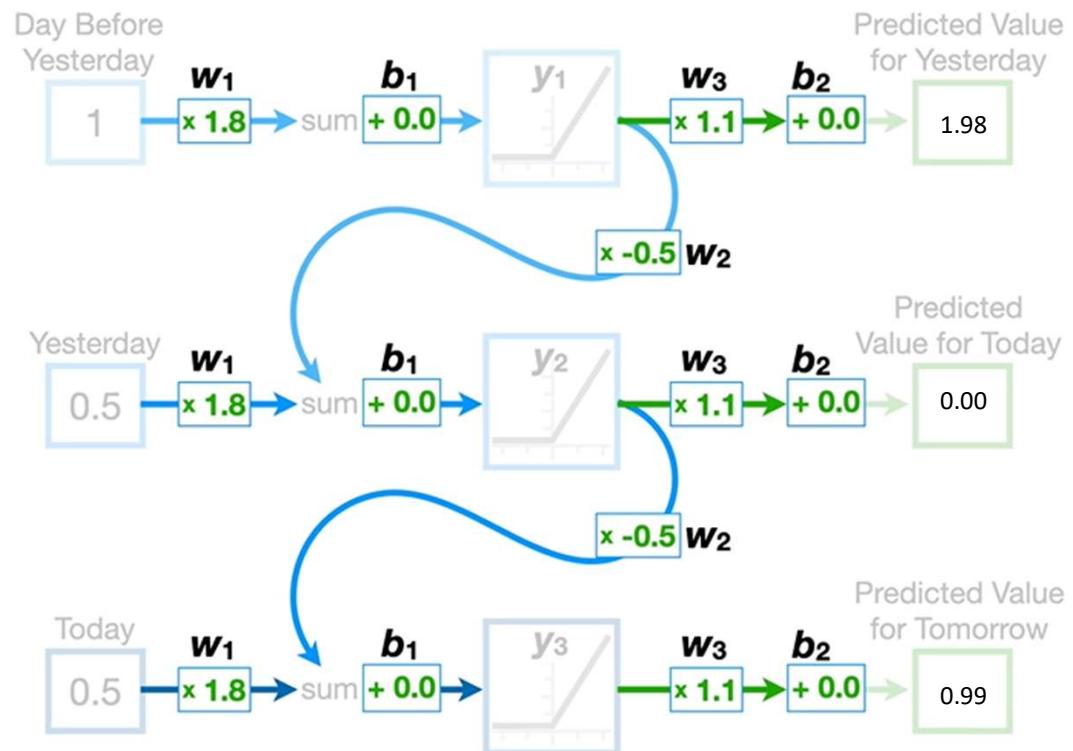
- **RNN**
- (engl. "compressed" or
"collapsed" representation)
- ("komprimierte" Darstellung)



Rechnen mit RNNs

Rekurrente Neuronale Netze (RNNs)

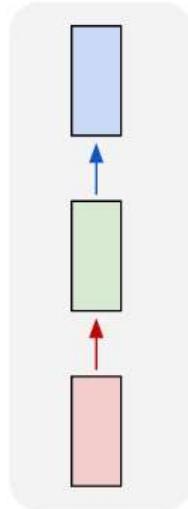
- Rechenbeispiel
- Nehmen wir an, wir haben die Gewichte des RNNs schon optimal bestimmt
- Rechnen wir den vorausgesagten Wert für morgen aus



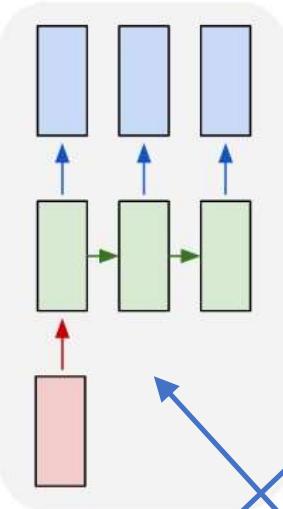
Rechnen mit RNNs

Rekurrente Neuronale Netze (RNNs)

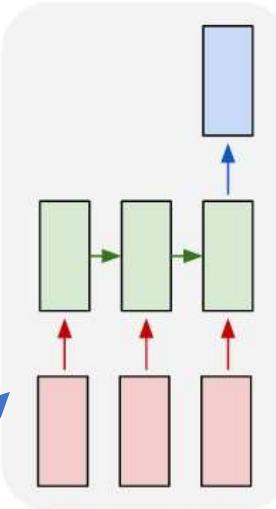
one to one



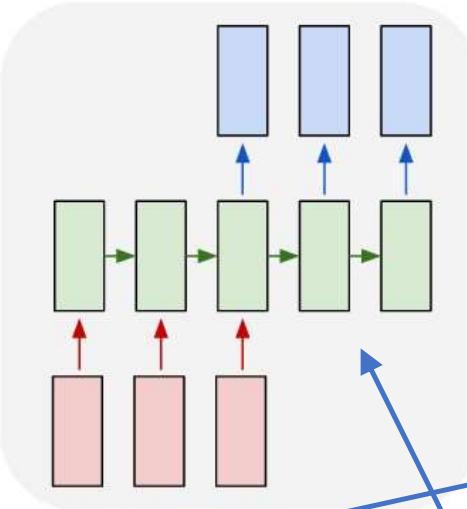
one to many



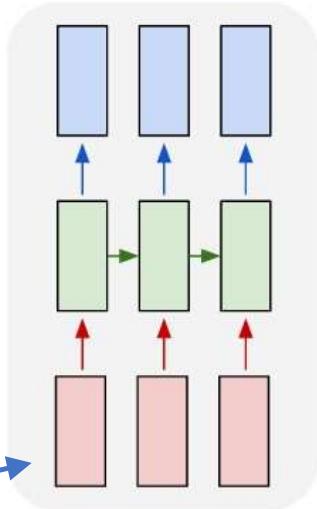
many to one



many to many



many to many



- e.g. sentiment classification, music generation, name entity recognition, language translation

Warum nutzen wir “Vanilla RNNs” selten?

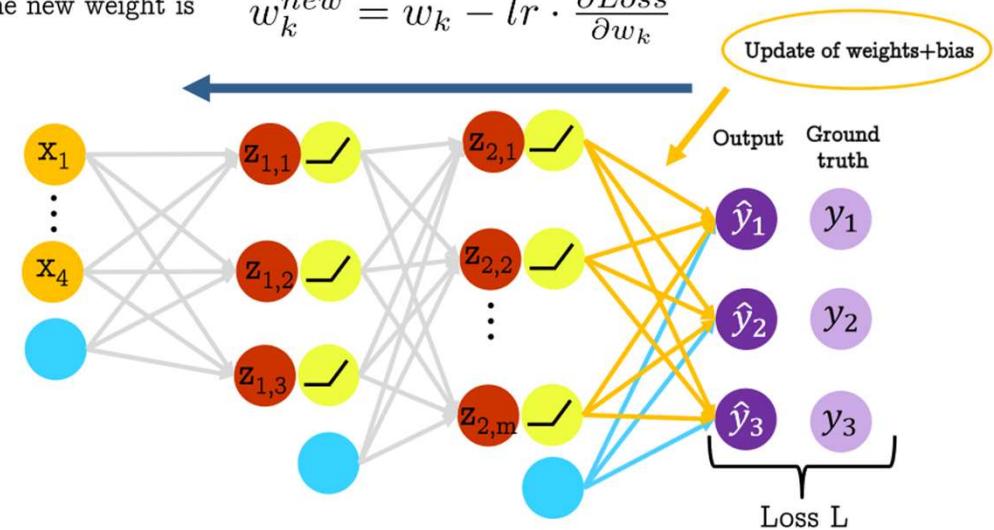
Rekurrente Neuronale Netze (RNNs)

- Ihr erinnert euch daran, wie die Gewichte schrittweise angepasst werden
- Indem man die partielle Ableitung des Losses nach dem Gewicht nimmt



Backpropagation – gradient descent optimizer

- ⇒ The direction for changing the parameters is calculated by taking the **partial derivative of the loss** with respect to the weight $\frac{\partial \text{Loss}}{\partial w_k}$
- ⇒ The new weight is $w_k^{new} = w_k - lr \cdot \frac{\partial \text{Loss}}{\partial w_k}$



Warum nutzen wir “Vanilla RNNs” selten?

Rekurrente Neuronale Netze (RNNs)

- Ihr erinnert euch daran, wie die Gewichte schrittweise angepasst werden
- Indem man die partielle Ableitung des Losses nach dem Gewicht nimmt
- Um sie zu berechnen, nehmen wir die Kettenregel
- Wir multiplizieren also durch das Netz durch : (wenn es sehr tief ist, bekommen wir evtl. Probleme)

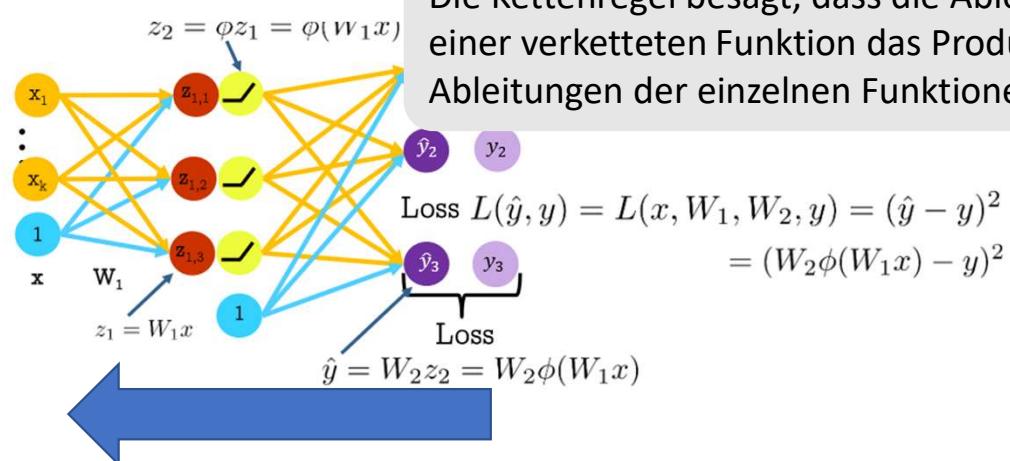
For computing the gradients $\frac{\partial L}{\partial W_1}$, $\frac{\partial L}{\partial W_2}$ we apply the chain rule and compute $\frac{\partial \text{layer_output}}{\partial \text{layer_input}}$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W_2} = 2(\hat{y} - y)(\phi(z_1))^T$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_2} \frac{\partial z_2}{\partial z_1} \frac{\partial z_1}{\partial W_1} = 2(\hat{y} - y)W_2^T \frac{\partial \phi(z_1)}{\partial z_1} x^T$$

$$W_1^{\text{new}} = W_1 - lr \cdot \frac{\partial L}{\partial W_1}$$

$$W_2^{\text{new}} = W_2 - lr \cdot \frac{\partial L}{\partial W_2}$$



Die Kettenregel besagt, dass die Ableitung einer verketteten Funktion das Produkt der Ableitungen der einzelnen Funktionen ist.

Warum nutzen wir “Vanilla RNNs” selten?

Rekurrente Neuronale Netze (RNNs)

- Bei RNNs bekommen wir noch ein Problem...
- Wir haben eine „Verkettung“ über unsere gesamte Länge der Eingabe
- Wenn wir die Kettenregel anwenden, multiplizierst du den Gradienten des aktuellen Zeitschritts mit dem Gradienten des vorherigen Zeitschritts, und so weiter, für alle Zeitschritte.
- Dies führt zu einer n-fachen Multiplikation der Gradienten, wobei n die Anzahl der Zeitschritte ist.
- $2^{50} = 1.125.899.906.842.624$
- $0.5^{50} = 0,00000000000000088817$



Die Kettenregel besagt, dass die Ableitung einer verketteten Funktion das Produkt der Ableitungen der einzelnen Funktionen ist.

Vanishing/exploding gradient

- Verschwindende und explodierende Gradienten häufig in RNNs
- Grund : schwierig, langfristige Abhängigkeiten zu erfassen, da der multiplikative Gradient mit der Anzahl der Schichten exponentiell ab- oder zunehmen kann
- Um das Problem zu lösen: Gates, die in der Regel einen genau definierten Zweck haben.

Und dann kam der Sepp...

Long Short-Term Memory (LSTM)



Quelle: <https://ajpod.de/podcast-archive/133>

LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter

Fakultät für Informatik

Technische Universität München

80290 München, Germany

hochreit@informatik.tu-muenchen.de

<http://www7.informatik.tu-muenchen.de/~hochreit>

Jürgen Schmidhuber

IDSIA

CORSO ELVEZIA 36

6900 LUGANO, SWITZERLAND

juergen@idsia.ch

<http://www.idsia.ch/~juergen>

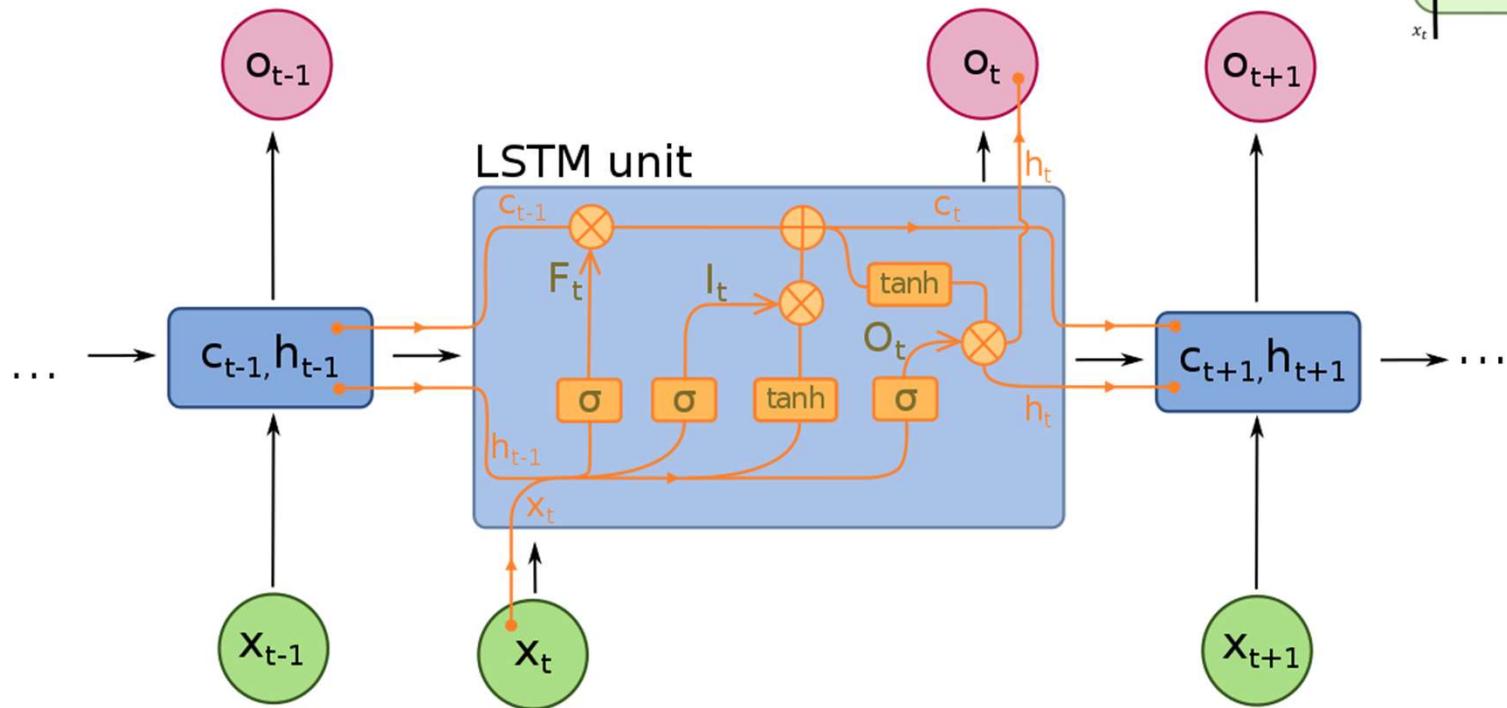
Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

1 INTRODUCTION

Die Idee von LSTMs

Long Short-Term Memory (LSTM)

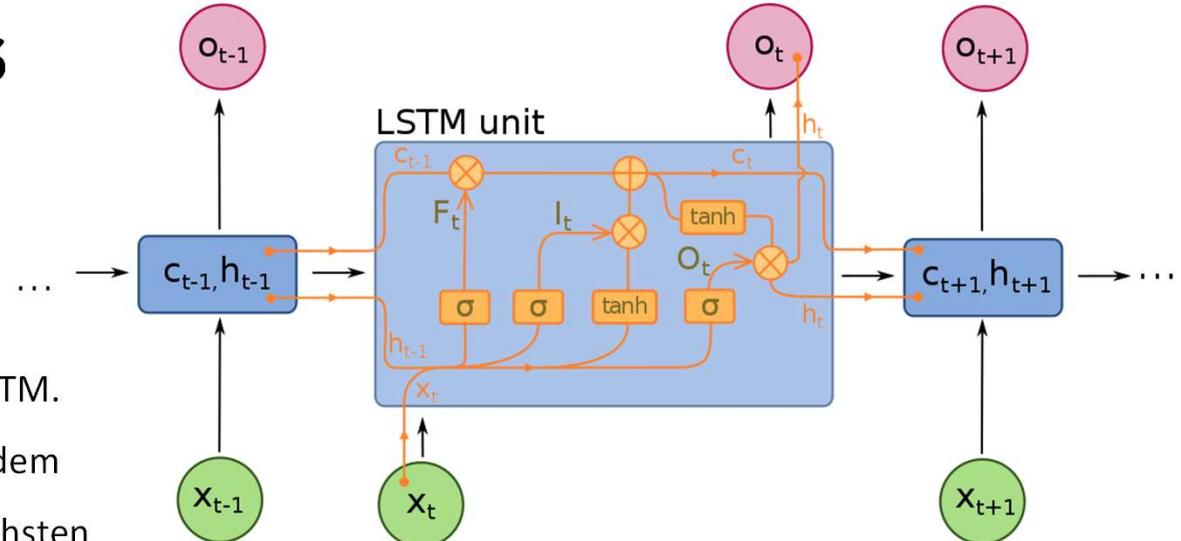


Quelle: https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg

Die Idee von LSTMs

Long Short-Term Memory (LSTM)

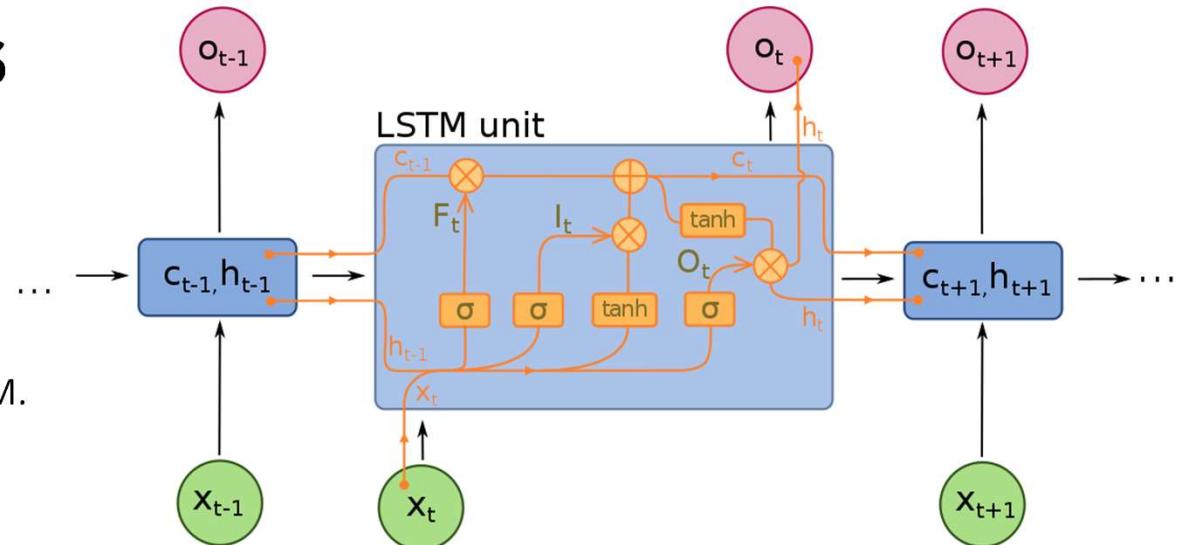
- Hidden State h_t :
 - Dies ist das "Kurzzeitgedächtnis" des LSTM.
 - Er wird von der aktuellen Eingabe und dem Zellzustand beeinflusst und wird im nächsten Zeitschritt an die LSTM-Einheit weitergegeben und auch für die Ausgabe des aktuellen Zeitschritts verwendet.
 - Der Hidden State wird in der Regel verwendet, wenn wir das LSTM für Vorhersagen nach jedem Zeitschritt nutzen wollen.



Die Idee von LSTMs

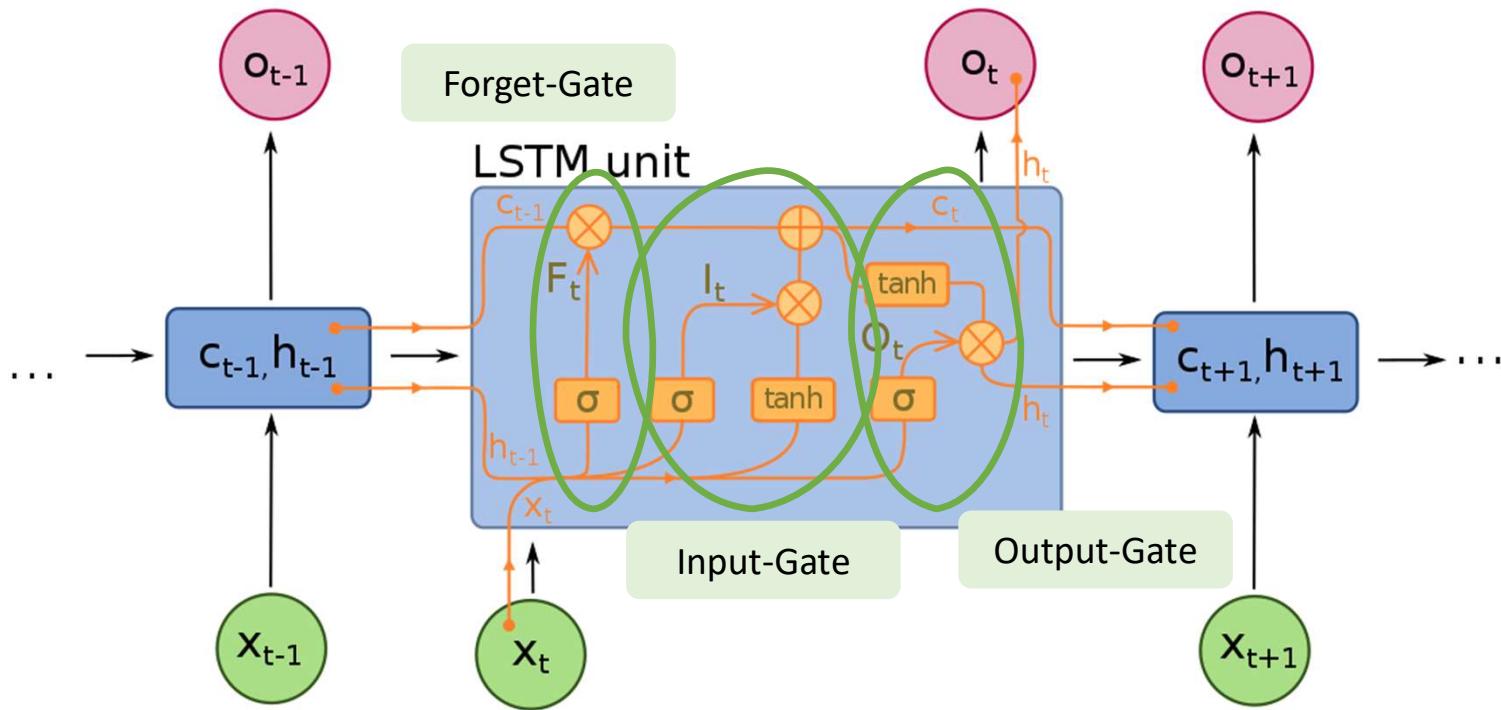
Long Short-Term Memory (LSTM)

- Zellzustand (Cell State) c_t :
 - Dies ist der "Langzeitspeicher" des LSTM.
 - Er durchläuft alle LSTM-Einheiten und überträgt Informationen aus früheren Zeitschritten in spätere Zeitschritte.
 - Die Informationen im Zellstatus können über Gates, die während des Trainings gelernt werden, hinzugefügt oder entfernt werden.
 - Diese Gates entscheiden, welche Informationen durchgelassen werden, und sie sind der Grund, warum das LSTM gut darin ist, langfristige Abhängigkeiten zu lernen.



Die Gates des LSTMs

Long Short-Term Memory (LSTM)



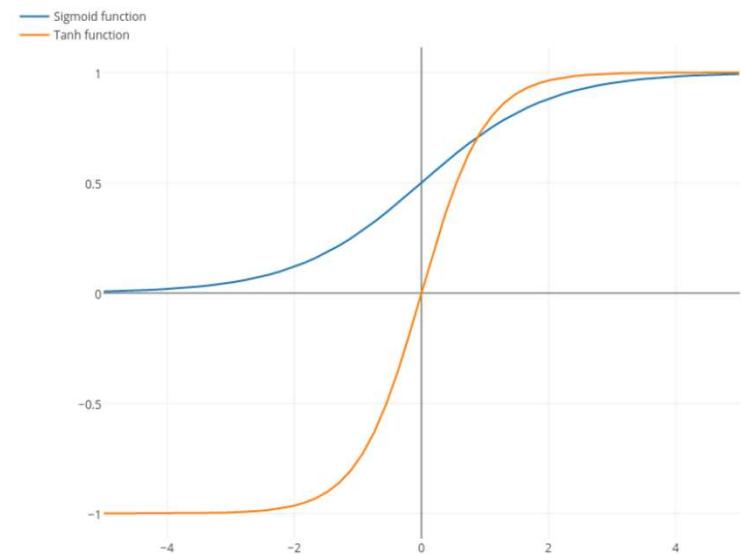
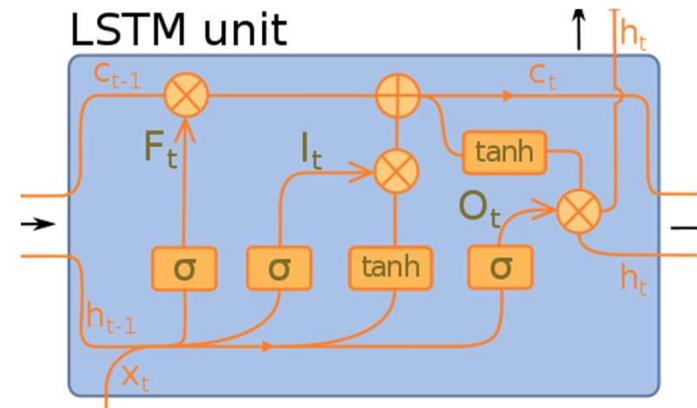
Quelle: https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg

79

Die Gates des LSTMs

Long Short-Term Memory (LSTM)

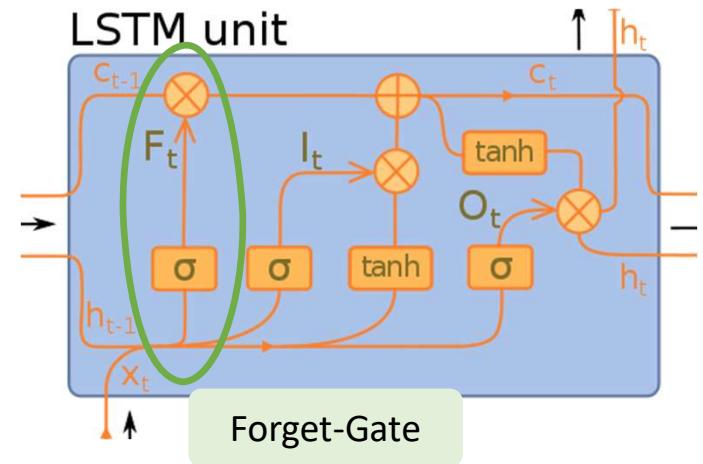
- **Die Sigmoid-Funktion:** Diese Funktion gibt Werte im Bereich zwischen 0 und 1 zurück. Daher ist sie besonders nützlich, wenn die Ausgabe des Netzwerks eine Wahrscheinlichkeit oder ein Prozentsatz darstellen soll.
- **Die Tanh-Funktion (Hyperbolische Tangente):** Die Tanh-Funktion gibt Werte im Bereich zwischen -1 und 1 zurück. Dadurch ist sie oft nützlich, wenn die Ausgabe um Null zentriert sein soll.



Das Forget Gate

Long Short-Term Memory (LSTM)

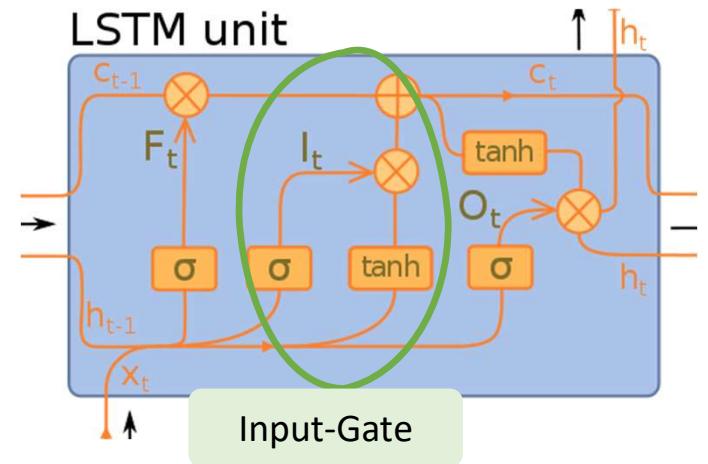
- Dieses Gate entscheidet, welche Informationen aus dem Zellzustand vergessen werden sollen.
- Es verwendet eine Sigmoid-Aktivierungsfunktion, so dass seine Ausgabe zwischen 0 und 1 liegt.
- Werte nahe 0 bedeuten, dass mehr Informationen vergessen werden, während Werte nahe 1 bedeuten, dass mehr Informationen beibehalten werden.



Das Input Gate

Long Short-Term Memory (LSTM)

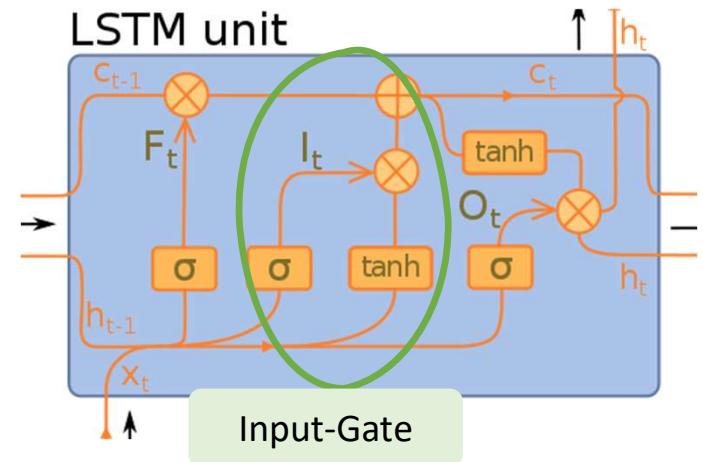
- Input-Gate:
 - Hier wird eine Sigmoid-Aktivierungsfunktion verwendet.
 - Das Gate erhält die aktuelle Eingabe und den vorherigen Hidden State (aus dem vorherigen Zeitschritt) und gibt Werte zwischen 0 und 1 aus.
 - Diese Werte stellen Gewichte dar, die angeben, wie wichtig jede Information für den aktuellen Zellzustand ist.
 - Werte nahe 0 bedeuten, dass die entsprechenden Informationen als unwichtig eingestuft und daher größtenteils ignoriert werden, während Werte nahe 1 bedeuten, dass die Informationen als wichtig angesehen und daher stärker berücksichtigt werden.



Das Input Gate

Long Short-Term Memory (LSTM)

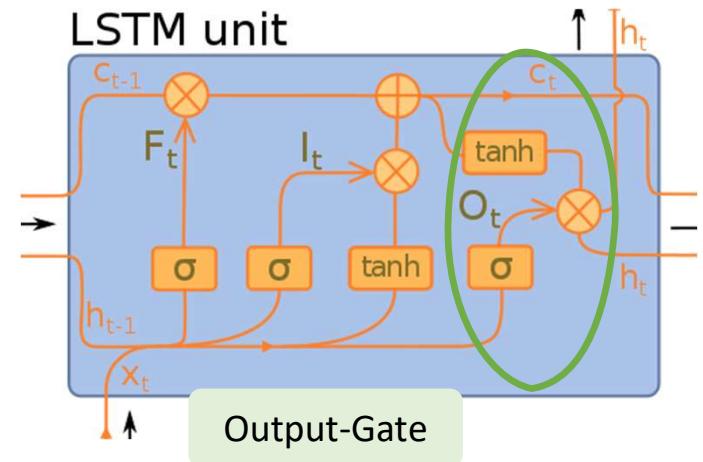
- Zustandskandidat:
 - Hier wird eine Tanh-Aktivierungsfunktion verwendet.
 - Ähnlich wie das Input-Gate erhält der Zustandskandidat die aktuelle Eingabe und den vorherigen Hidden State, aber die Tanh-Funktion skaliert die Ausgabe auf Werte zwischen -1 und 1.
 - Dieser skalierte Output wird dann mit dem Output des Input-Gates multipliziert, um eine modifizierte Version der Eingabe zu erzeugen, die möglicherweise in den Zellzustand aufgenommen wird.



Das Output Gate

Long Short-Term Memory (LSTM)

- Output Gate:
 - Dieses Gate entscheidet, welche Teile des Zellzustands in den hidden state und zur Ausgabe übertragen werden sollen.
 - Es verwendet eine Sigmoid-Aktivierungsfunktion auf die Eingabe und den vorherigen hidden state, während eine Tanh-Aktivierung auf den Zellzustand angewendet wird.



LSTM Vorteile

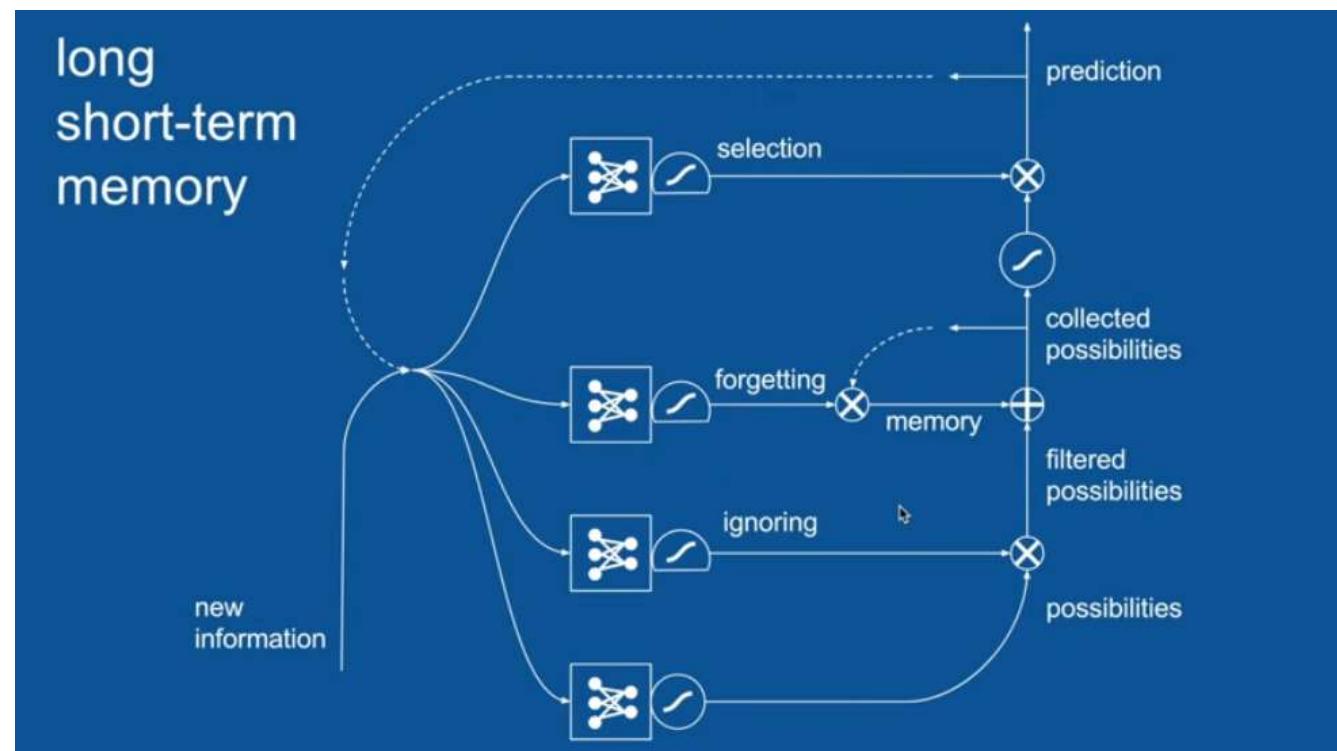
Long Short-Term Memory (LSTM)

- **Lösen des Verschwindenden Gradientenproblems:** LSTMs lösen dieses Problem durch die Einführung des Zellzustands, einer Art von "Informationsträger", der es ermöglicht, Information über längere Zeiträume hinweg zu behalten.
- **Gating-Mechanismen:** Die Gates ermöglichen es den LSTMs, relevante Informationen zu speichern und unwichtige Informationen zu vergessen, was zu einem effektiveren Lernen führt.
- **Bessere Leistung bei vielen Aufgaben:** In der Praxis haben LSTMs oft eine bessere Leistung als einfache RNNs bei vielen Arten von Aufgaben gezeigt, insbesondere solchen, die das Lernen von langfristigen Abhängigkeiten erfordern, wie z. B. Spracherkennung, maschinelles Übersetzen, Sprachgenerierung und viele andere.
- **Flexibilität:** LSTMs können komplexe Muster und Strukturen erkennen

LSTM-Anwendungen

Long Short-Term Memory (LSTM)

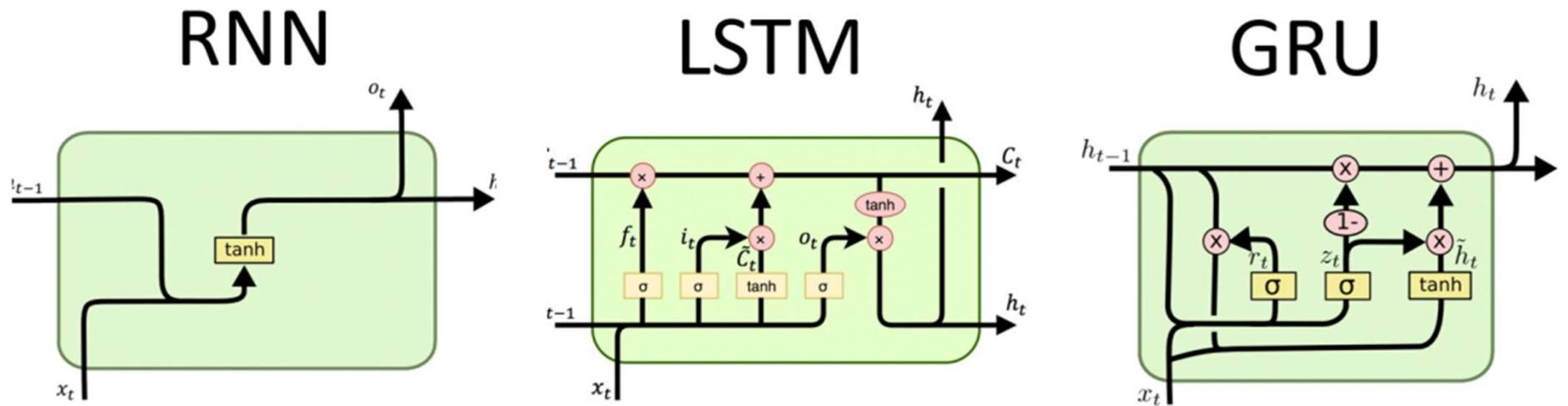
- Beispiel für LSTM
Natural Language Processing



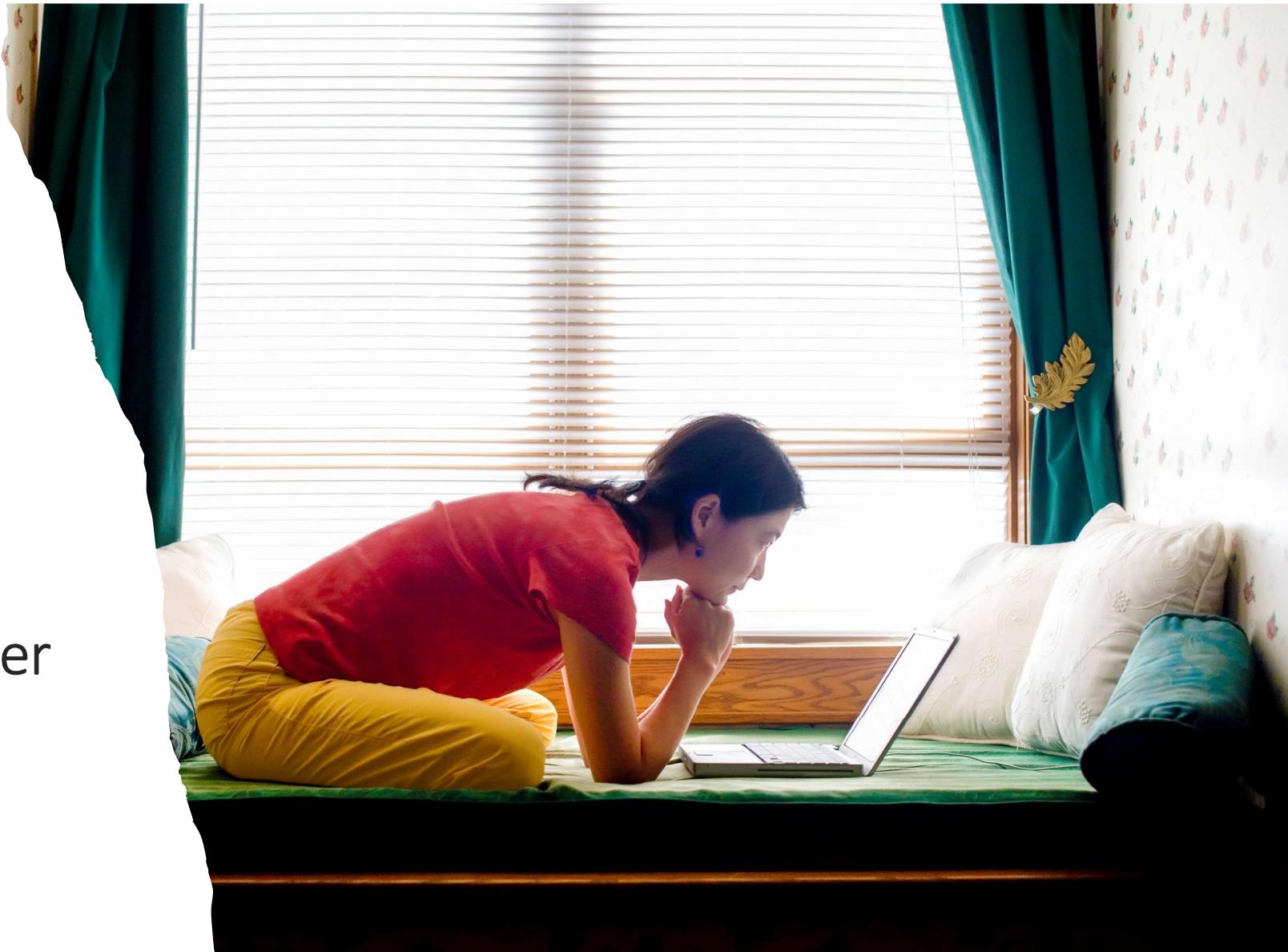
LSTM vs GRU

Long Short-Term Memory (LSTM)

- Ein Gated Recurrent Unit (GRU) ist eine andere Art RNN. GRUs wurden 2014 eingeführt als eine vereinfachte Alternative zu Long Short-Term Memory Units (LSTMs), die einen ähnlichen Zweck erfüllen.



Ab ins Jupyter
Notebook



Inhalte – Was haben wir gemacht?

Tag 1

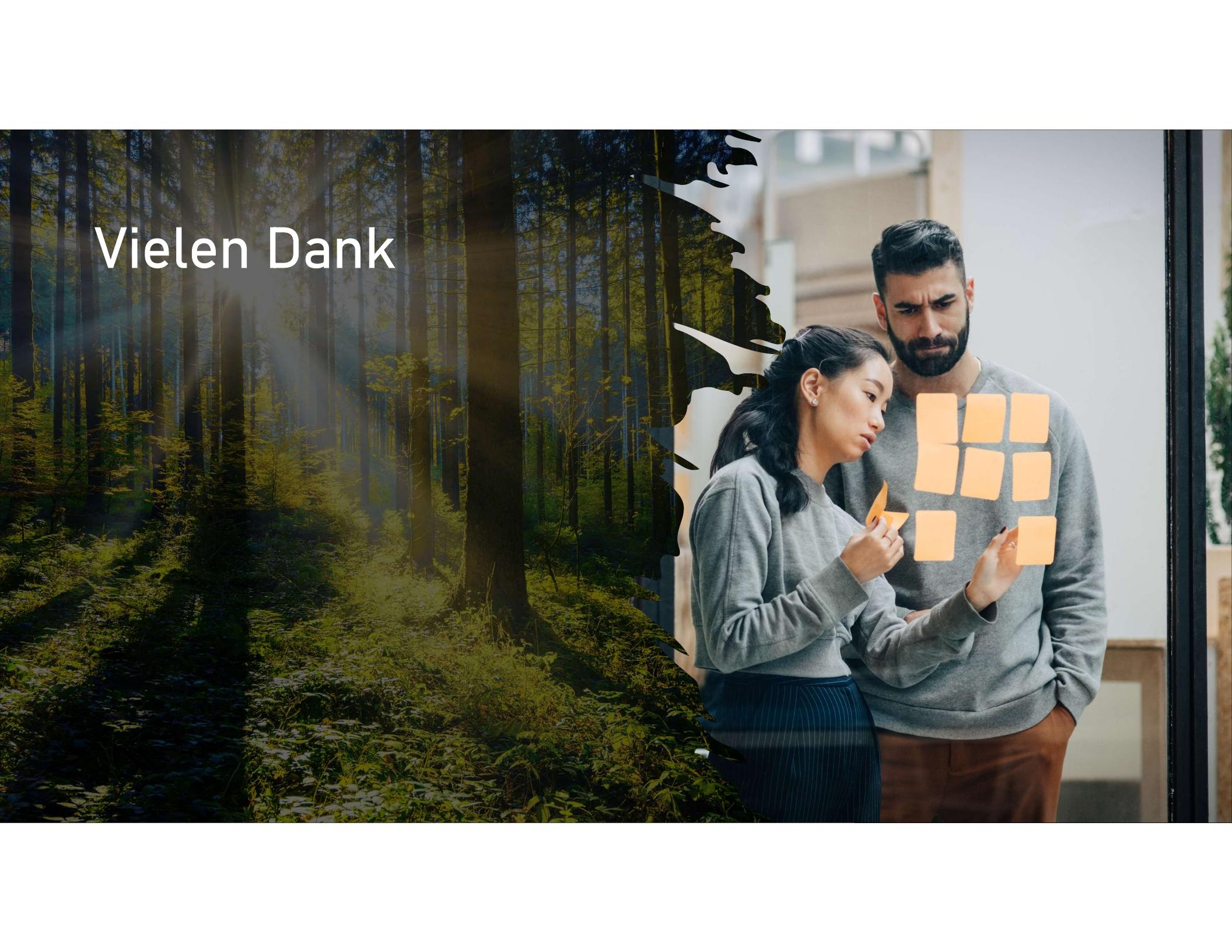
1. Einführung in Zeitreihendaten in Python
2. Zeitreihen und ihre Merkmale visualisieren
3. Zeitreihen vorhersagen (Statistik I): Exponentielle Glättung und Holt-Winters
4. Zeitreihen vorhersagen (Statistik II): ARIMA-Modelle

Tag 2

5. Einblick in andere Zeitreihenmodelle
6. Machine Learning für Zeitreihen: Überblick, Vorbereitung und Vorhersagen
7. Machine Learning für Zeitreihen: Clustering und Klassifikation
8. Deep Learning für Zeitreihen

Feedback



A composite image. On the left, a dense forest with tall trees and sunlight filtering through the canopy. On the right, two people, a man and a woman, are standing in an office setting. They are looking at a whiteboard covered with several orange sticky notes arranged in a grid pattern. The man has a beard and is wearing a grey sweatshirt, while the woman is wearing a grey sweater and dark pants.

Vielen Dank