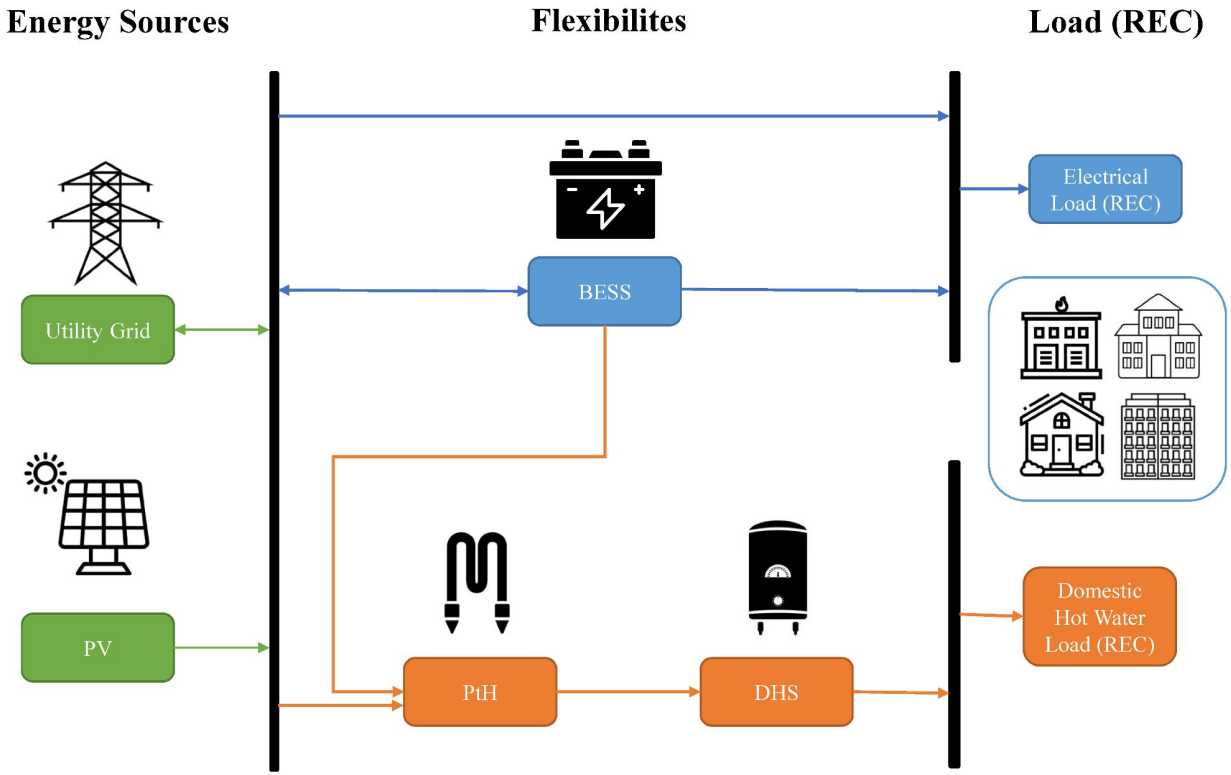
Workshop

Prognosen von Zeitreihen mit Machine Learning

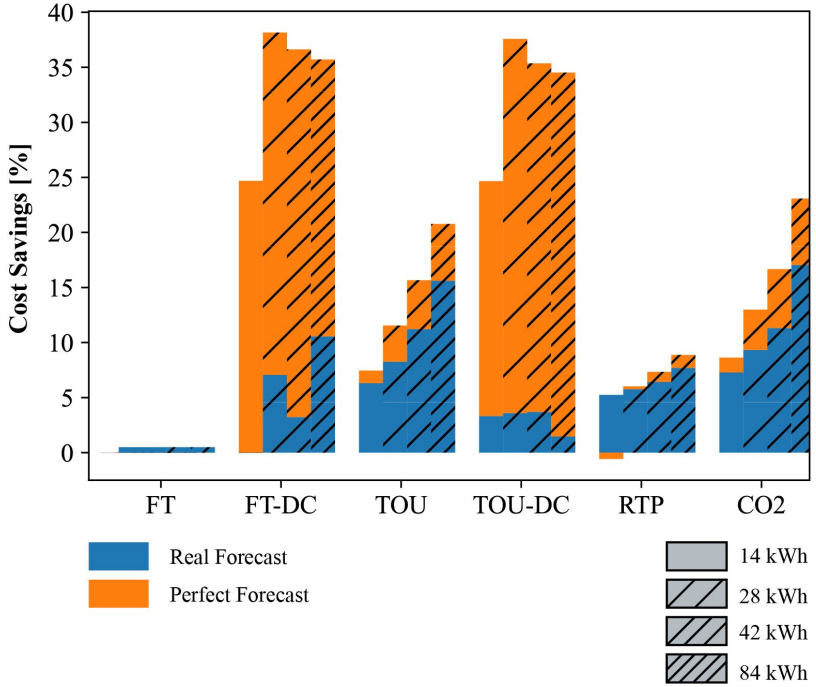
Nikolaus Houben
houben@eeg.tuwien.ac.at

Motivation


Energy Sources
Flexibilites
Load (REC)
Utility Grid
PV
BESS
PtH
DHS
Electrical Load (REC)
Domestic Hot Water Load (REC)

Motivation

- Zeitreihen sind überall und wer sie prognostizieren kann, macht Geld
- Beispiel: Microgrids
  - Vergleich eines Standardregler mit einem Smart Grid Controller
  - Mit fehlerfreien Prognosen sind Einsparungen von bis zu 35% möglich.
  - Sonst nur ca 15%



Der Plan für Heute

01

Basics

Vortrag
10:00-10:15

02

Data Exploration

Code-Along
10:15-10:45

03

Machine Learning

Vortrag
10:45-12:30

04

Mittagessen

-
12:30-13:30

05

Pytorch & Sklearn

Code-Along
13:30- 15:00

06

Recurrent Networks

Vortrag
15:20 - 16:00

07

Transfer Learning

Vortrag + Code
16:00-16:45

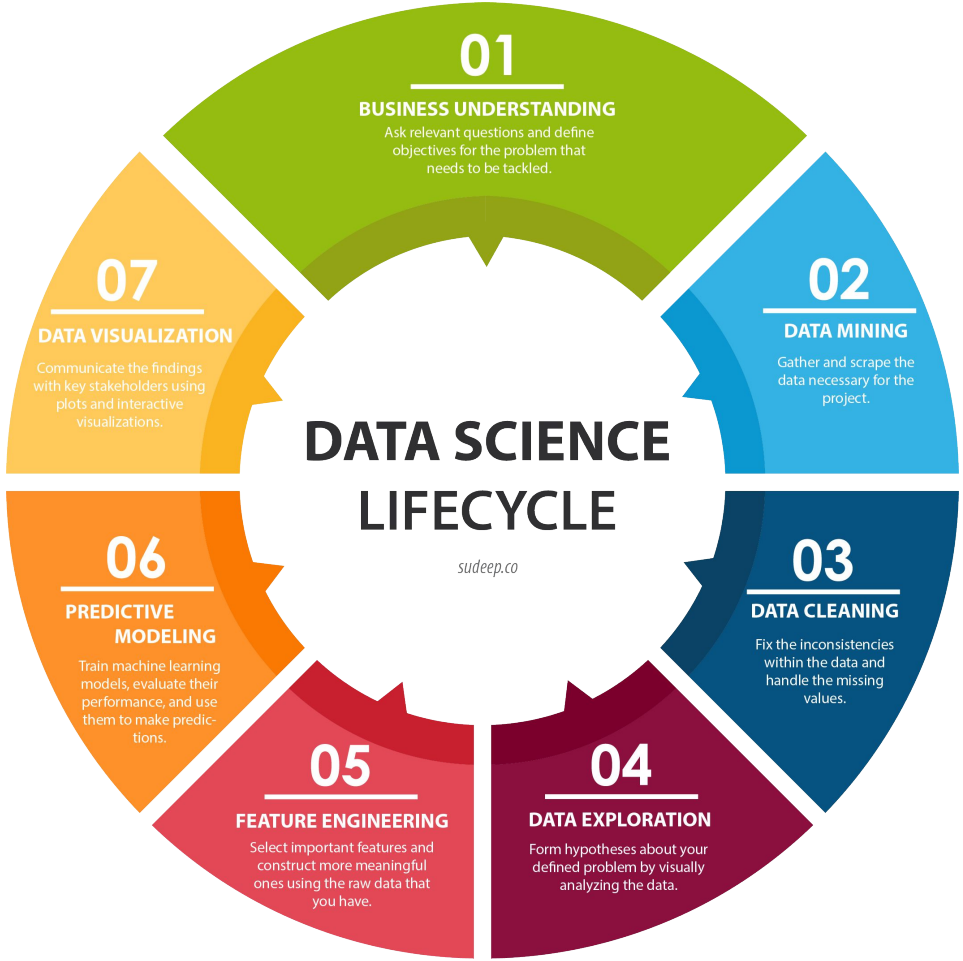
08

Zusammenfassung

Code-Alone
16:45-17:00

01 Basics: Data Science

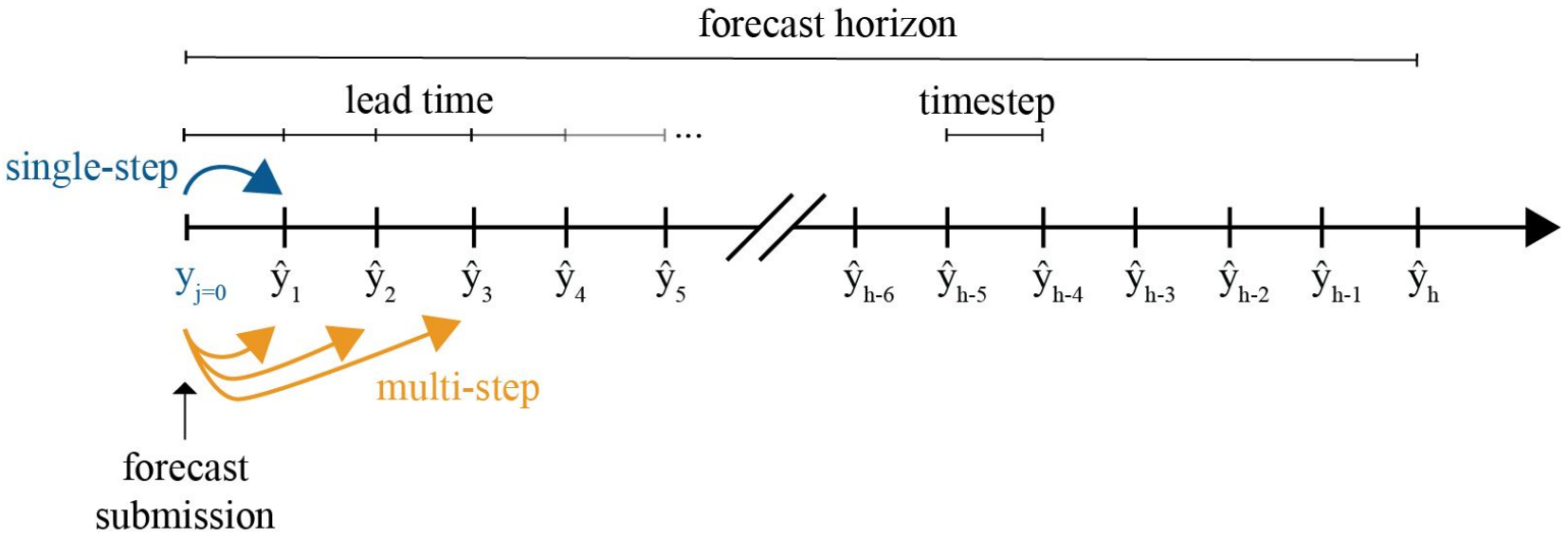
Vortrag



Source: https://www.sudeep.co/data-science/2018/02/09/Understanding-the-Data-Science-Lifecycle.html

Zeitreihen

- Der “Business Context” stellt Anforderungen an das Prognosemodell:



Vier Wichtige Konzepte

1 Stationarität

Ist das Mittel und die Varianz der Zeitreihe über mehrere Zeitschritte konstant?

2 Autokorrelation

Korrelieren die Werte der Zeitreihe untereinander?

3 Periodizität

Gibt es Abläufe die sich in regelmäßigen Abständen wiederholen?

4 Rauschen

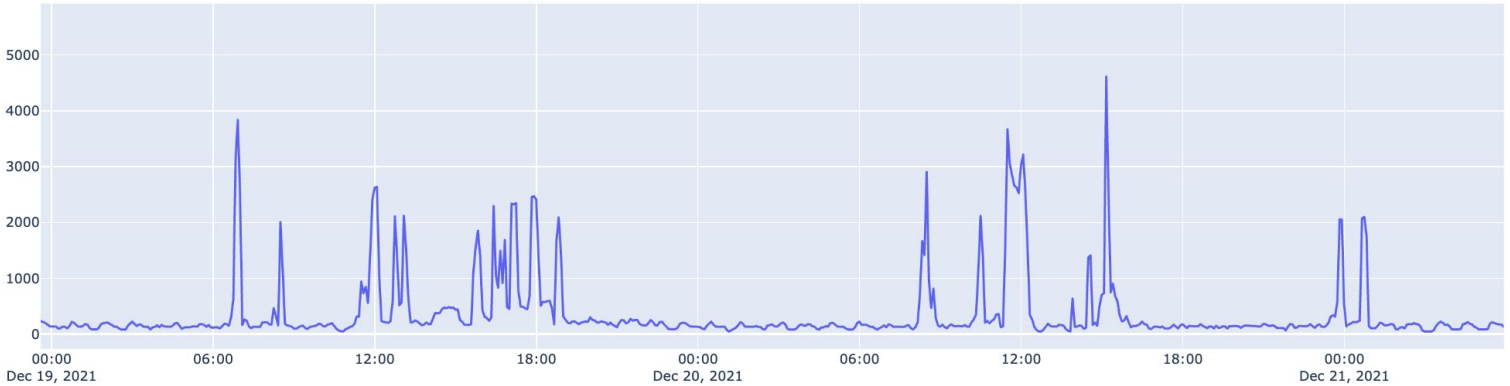
Gibt exogene “Gründe” für gewisse Abläufe?

02

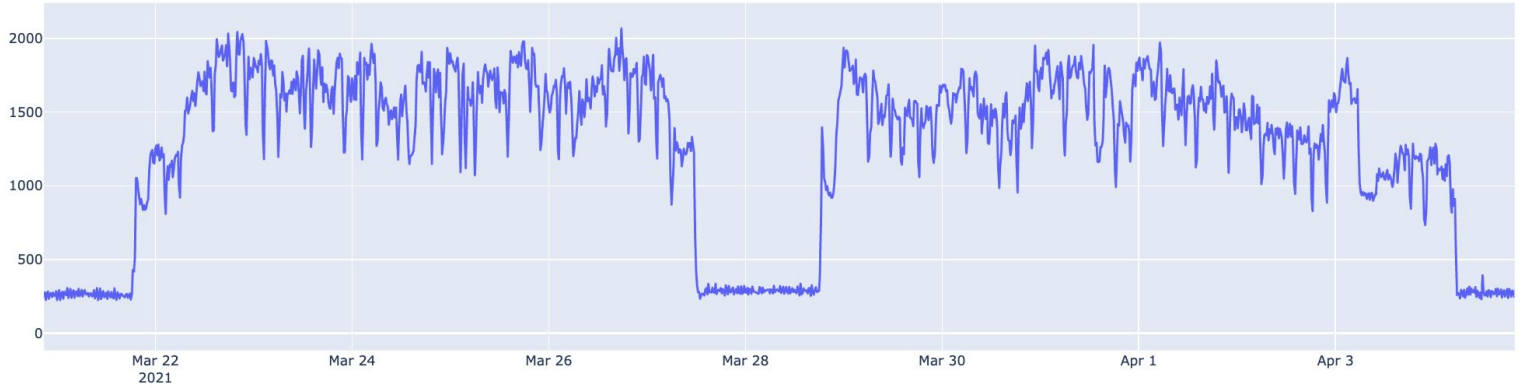
Erkundung der Daten

Code-Along

Datenlage



Datensatz #1



Datensatz #2

Notebook: shorturl.at/gPSU3

↣ Google Colab

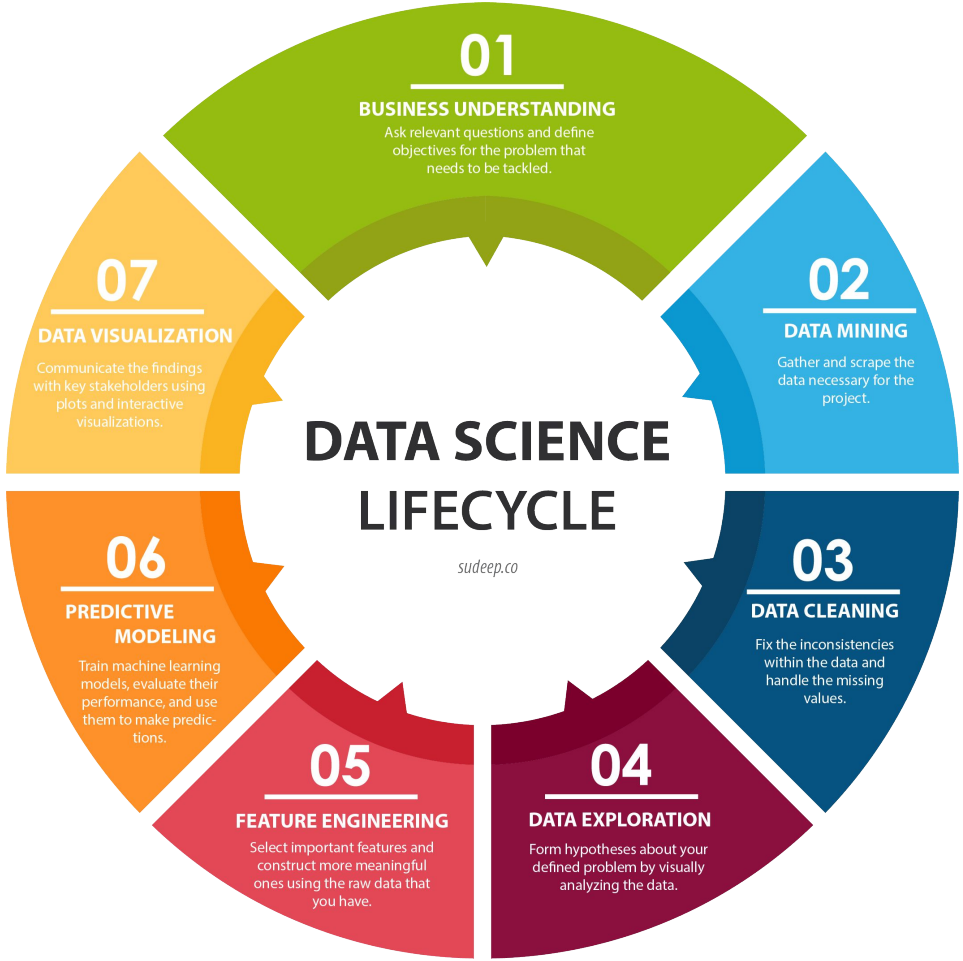
1. Was sind das für Daten?
2. Welche besonderen Merkmale könnt ihr in den Daten finden?
3. Welche Unterschiede gibt es zwischen den beiden Datensätzen?

03 Machine Learning

Vortrag

If **Data** is the new **oil** of the **21st century** then **machine learning** is the new **combustion engine**

Lösung: shorturl.at/JRX26



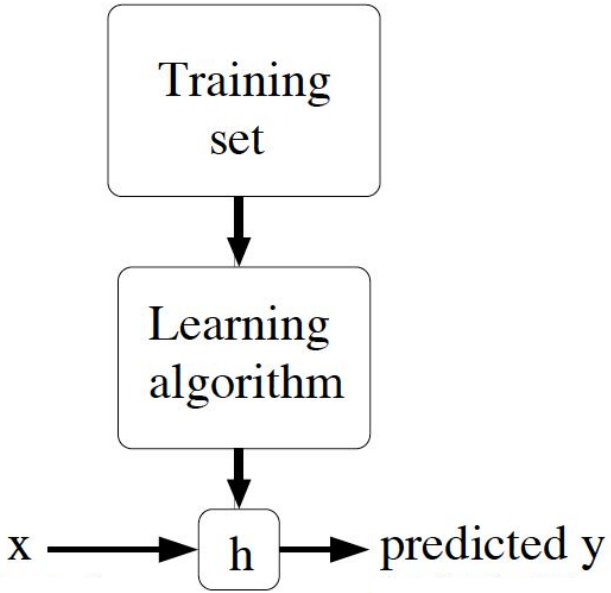
Source: https://www.sudeep.co/data-science/2018/02/09/Understanding-the-Data-Science-Lifecycle.html

Was muss man können um anzufangen?

1. *Mathematik*: Grundrechnungsarten, ein bisschen Lineare Algebra, Funktionen differenzieren (Kettenregel!);
2. *Informatik*: Python, und insbesondere Loops!
3. *(Wahrscheinlichkeitstheorie:* Schätzer, Satz von Bayes) - Nicht unmittelbar relevant aber später dann

Die Idee von Machine Learning

- Idee:
  - Wir haben Input und Output Pärchen (Unsere Daten)
  - Wir teilen die Daten auf in Trainings Daten und Test Daten
  - Wir machen einen Ansatz für die Funktion z.B. ein Lineare Regression (“Learning Algorithm”)
  - Wir suchen die Parameter des Ansatzes in einem “Trainingsprozess”; Das ist das Lernen!
  - Wir messen die Performance auf den ungesehenen Test Daten und passen den Trainingsprozess an bis die Performance passt
  - Anpassen heißt: andere Algorithmen, Features, Hyperparameter, oder andere Tricks (später)
- Bezeichnungen:
  - Input = Features / Covariates / X
  - Output = Target (y)
  - Predicted y = Forecasts (ŷ)
  - Performance: ein Maß von Genauigkeit, das dem Business Context entspricht



Source: Lecture Notes CS229 Stanford; Andrew Ng

Der Startpunkt für Inputs/Features/X

2

Autokorrelation

Korrelieren die Werte der Zeitreihe untereinander?

Lag features:
Wert des letzten Zeitschritts
Wert vor genau einer Woche

3

Periodizität

Gibt es Abläufe die sich in regelmäßigen Abständen wiederholen?

Datetime Features:
Tag der Woche
Tageszeit

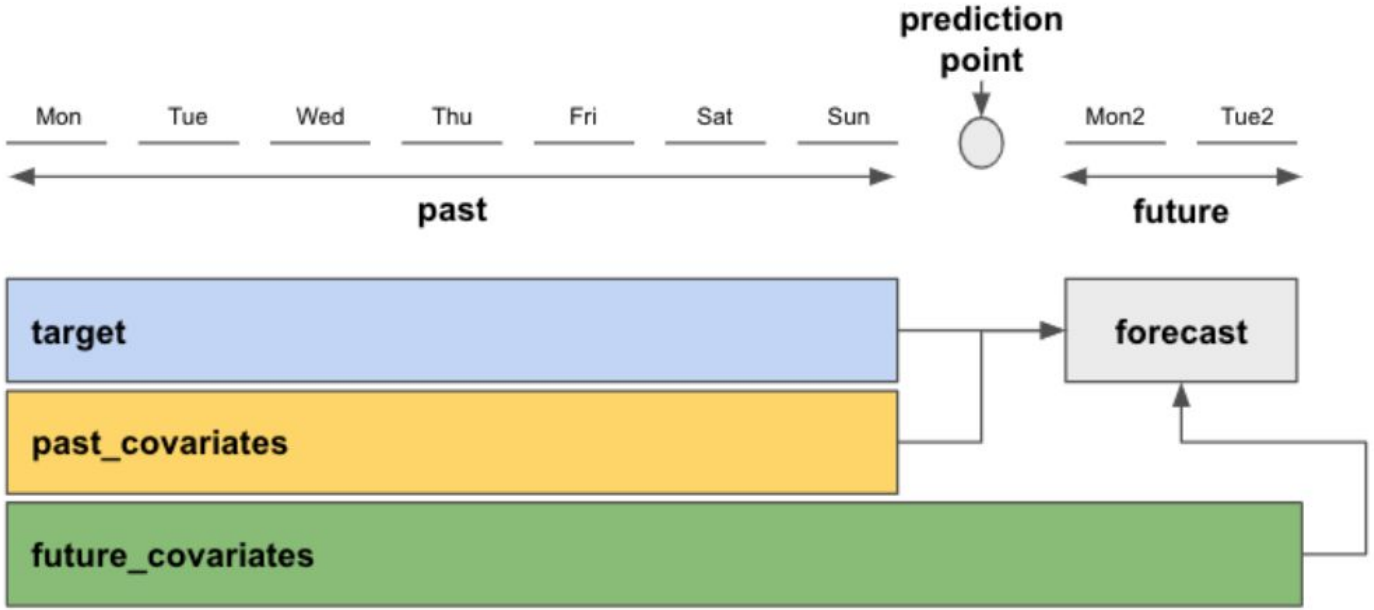
4

Rauschen

Gibt exogene “Gründe” für gewisse Abläufe?

Datetime Features:
Das Wetter (ev. als Forecast)
Tageszeit

Features und ihre Zeitspannen



Source: https://unit8co.github.io/darts/userguide/covariates.html

3.1

Algorithmen

Vortrag

(Lineare) Regression

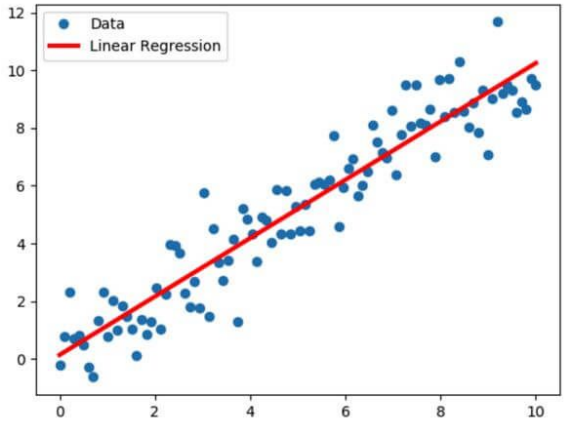
- Regression vs. Klassifikation: *Regression* heißt es wenn Targets Zahlenwerte sind
- Das ist bei Zeitreihenprognosen fast immer der Fall
- Wir wählen den Linearen Ansatz:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- Wir wählen eine *geeignete* Cost Function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

- Wir verwenden den Gradient Descent Algorithmus um die Parameter (Theta) zu lernen.



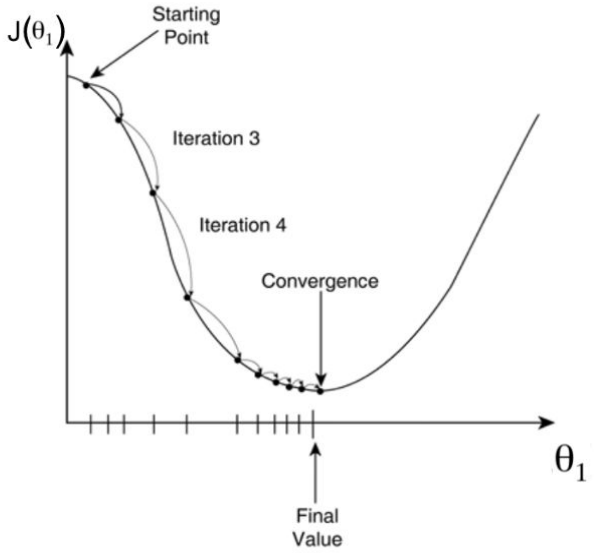
Andere Algorithmen

- Für stationäre, nichtlineare Zusammenhänge zwischen Features und Targets und WENIG “Tuning” Aufwand:
  - Support Vector Machine
  - Random Forest
  - XGBoost

- Für nicht-stationäre, nicht-lineare Zusammenhänge zwischen Features und Targets und VIEL “Tuning” Aufwand:
  - Recurrent Neural Networks (RNNs): LSTM, GRU
  - Transformers: Temporal Fusion Transformer
- Für Benchmarking geeignet (keine ML Algorithmen):
  - SARIMAX
  - Theta Method - Gute performance in dem M3-Wettbewerb
  - Prophet

Gradienten-basierende Algorithmen

- *Gradient Descent* ist einer der wichtigsten Bausteine von Neuronale Netzen



Cost Function – "One Half Mean Squared Error":

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Objective:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

Derivatives:

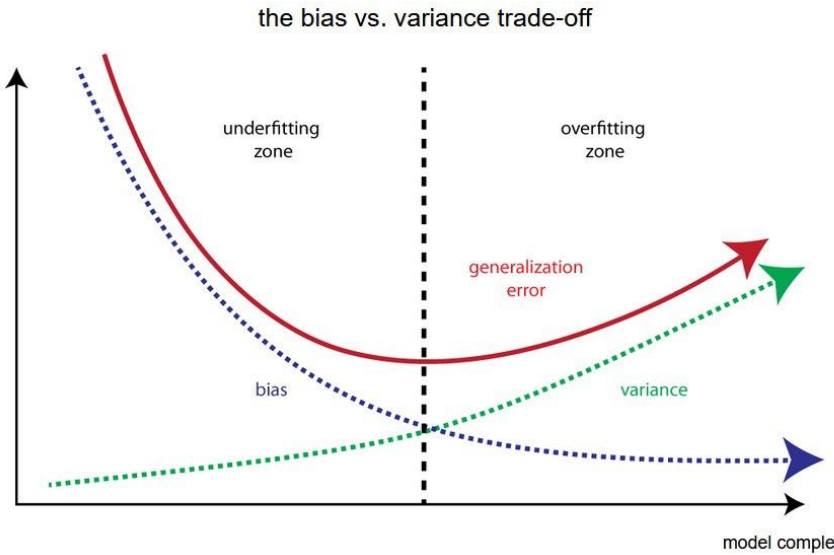
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x^{(i)}$$

Regularization

- Idee: Wir führen einen zusätzlichen Term in der “Cost Function” ein, der die Komplexität des Modells (z.B. die Anzahl der lernbaren Parameter) ausdrückt.
- Es soll somit eine gute Balance zwischen Over und underfitting gefunden werden.
- Anders gesagt: Wir reduzieren die “Variance” (weniger Parameter) indem wir ein bisschen mehr “Bias” dazugeben.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$



Source: https://towardsdatascience.com/bias-and-variance-but-what-are-they-really-ac539817e171

04 Sklearn

Code-Along

Notebook: shorturl.at/b0238

↣ Google Colab

1. Teile den Datensatz in ein Trainings und Test Set mit einem Verhältnis 80/20 auf.
2. Importiere das Lineare Regressions Model von Sklearn und rufe .fit auf, um es zu trainieren.
3. Vervollständige die “n_step_recursive_regression” Funktion!

04

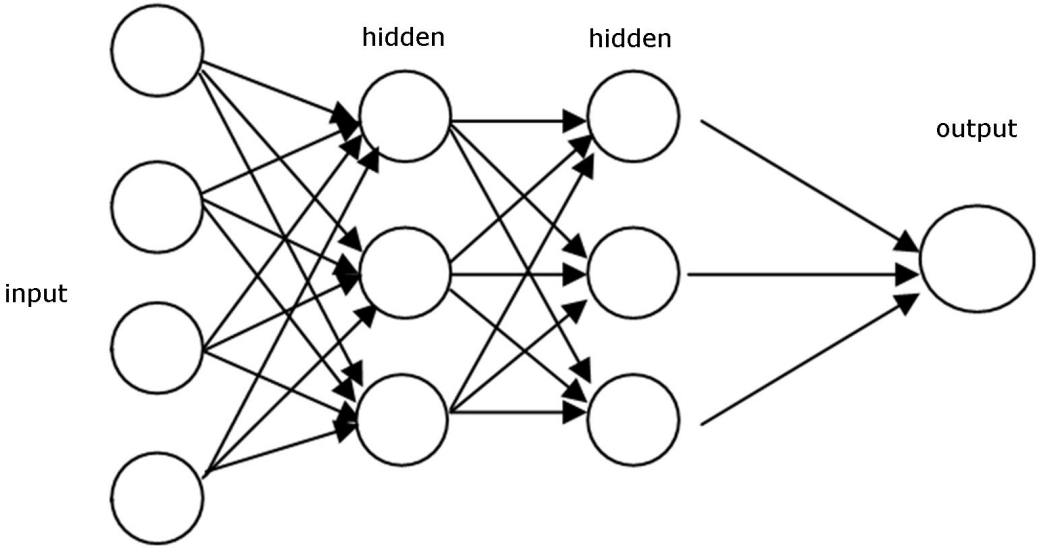
Neural Networks: Basics

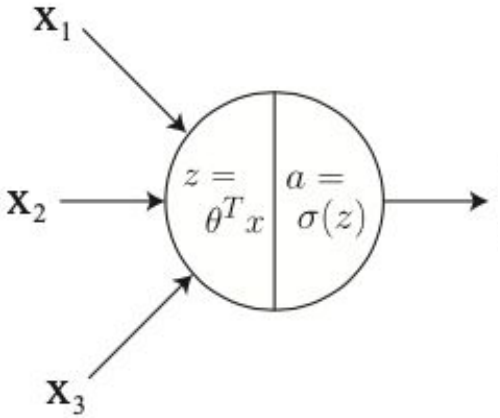
Vortrag

Lösung: shorturl.at/brVW0

Architektur: Neural Networks

- Ein Netz von Knoten: jeder Knoten beinhaltet eine lineare und nichtlineare Transformation
- Training/Lernen: Loop von Forward (links nach rechts) und Backward (rechts nach links)

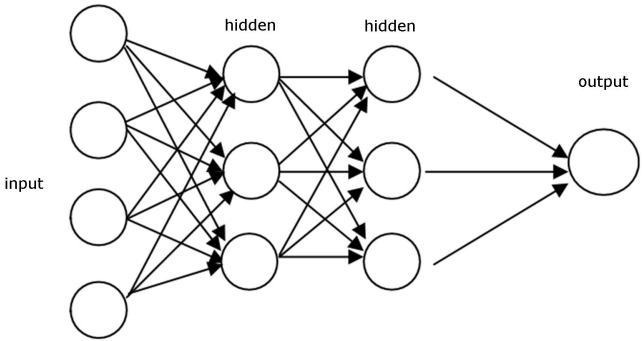


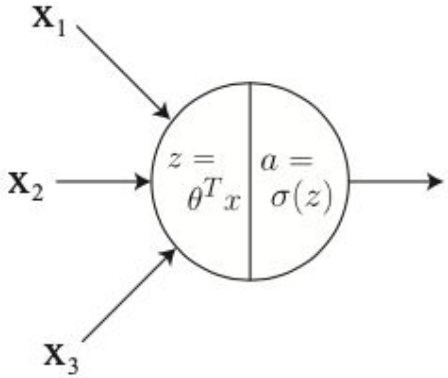


Foreward Pass (links nach rechts)

- Ziel: Berechnung der Cost Function
- Prozess: Serienschaltung von Knoten

1. Die Inputs werden durch den ersten Layer von Knoten linear und nichtlinear transformiert
2. Die Outputs des ersten Layers werden an den zweiten Layer weitergegeben
3. Und so weiter
4. Bis der letzte Layer erreicht ist, und (für Regression) alle Knoten auf nur noch einen Knoten aggregiert werden





Backward Pass (rechts nach links)

- Disclaimer: **Das komplizierteste Konzept in Deep Learning**
- Es gibt 2 Zutaten zu Backpropagation: Die Kettenregel, Lineare Algebra (Matrix*Vector)
- https://www.youtube.com/watch?v=9d2fwGjyb4M

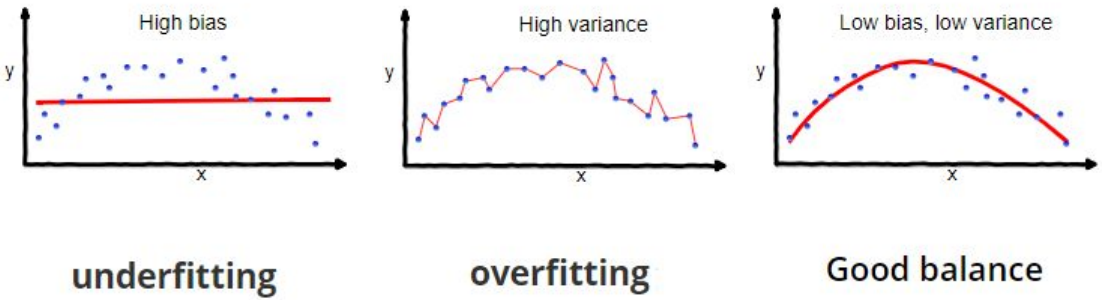
Source: https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0

Die Hebel der Modellierung

Viele Wege führen nach Rom. Doch beim Machine Learning gibt es eine klare Hierarchie die beachtet werden sollte, bevor man einen Weg einschlägt.

Das altbekannte “Bias-Variance Trade-off” liegt diesem Prozess zugrunde.

1. Daten (1. Menge und 2. Qualität)
2. Feature Engineering
3. Algorithmen / Architekturen
4. Hyperparameters: Regularization!
5. Loss Function
6. Bei Neural Networks - Tricks:
    a. Mehr Layer
    b. Early Stopping
    c. Drop-out
    d. RMSprop
    e. Batch normalization



Source: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

04 Pytorch

Code-Along

Notebook: shorturl.at/boqs1

↣ Google Colab

1. Vervollständige die Funktion "cumsum_per_day"
2. Skaliere die "train, val, und test" cumsum covariates
3. Führe das Backtesting des Model 3 für einen Zeitraum deiner Wahl aus.

04

Recurrent Neural Networks

Vortrag

Lösung: shorturl.at/eFTW6

Recurrent Neural Networks: Wieso?

Warum verwenden wir nicht weiterhin normale Neural Networks oder andere Regressionen mit Lags?

In einem Klassischen NN gibt es folgende Probleme:

- Die Reihenfolge der Input Features ist nicht explizit vorgegeben
- Die Gewichte die für einen Feature gelernt werden, können nicht für andere Features verwendet werden
- Unterschiedliche längen der Input und Output Vektoren pro Trainingsbeispiel sind nicht unterstützt

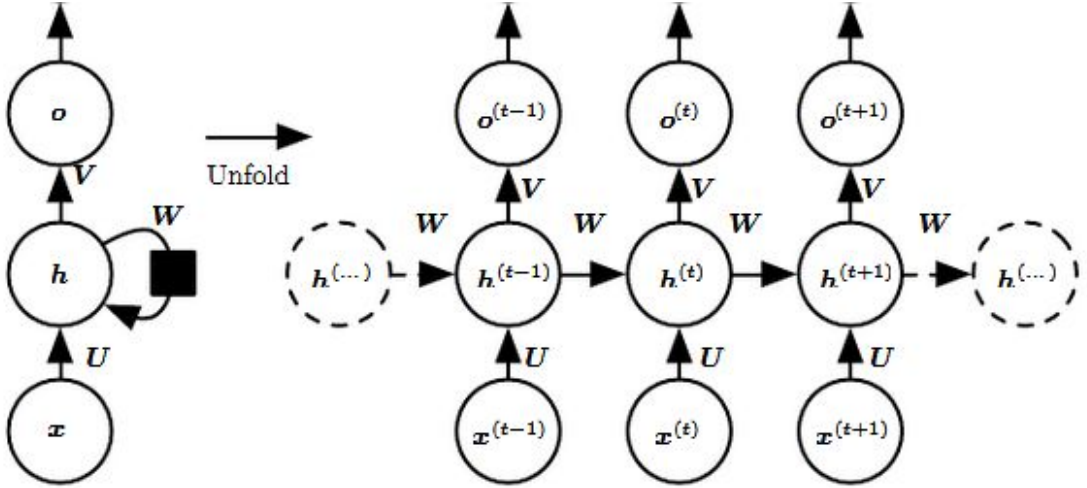
Recurrent Neural Networks werden in folgenden Bereichen mit großem Erfolg verwendet:

- Spracherkennung (Siri, Google Voice Search)
- Musikerkennung (Shazam)
- DNA Sequenzierung
- Zeitreihen Forecasting

Recurrent Neural Networks: Was?

- RNNs werden als rekurrent bezeichnet, da sie für jedes Element einer Sequenz dieselbe Aufgabe ausführen, wobei die Ausgabe von den vorherigen Berechnungen abhängt
- Zentrale Idee: Information von vorherigen Zeitschritten wird weitergegeben (Gedächtnis!!!)

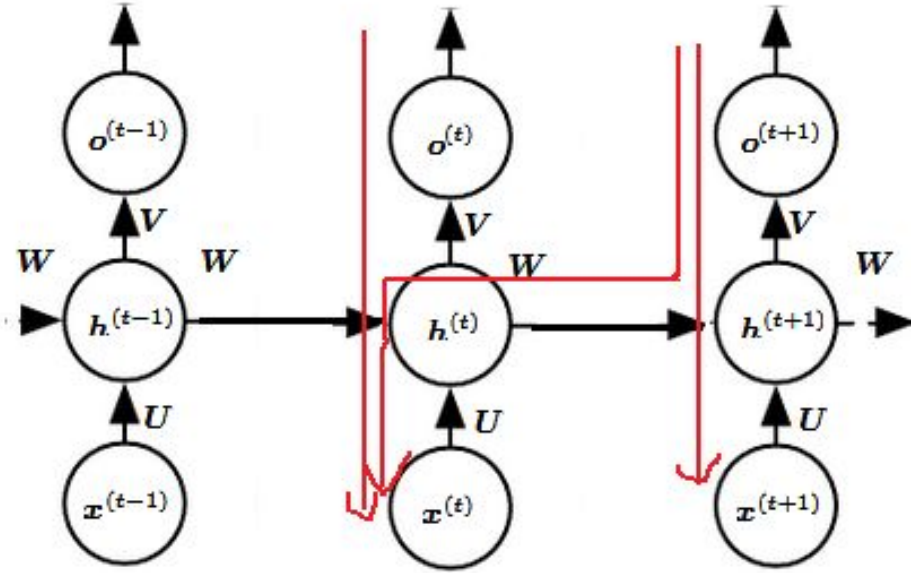
$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$



Source: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

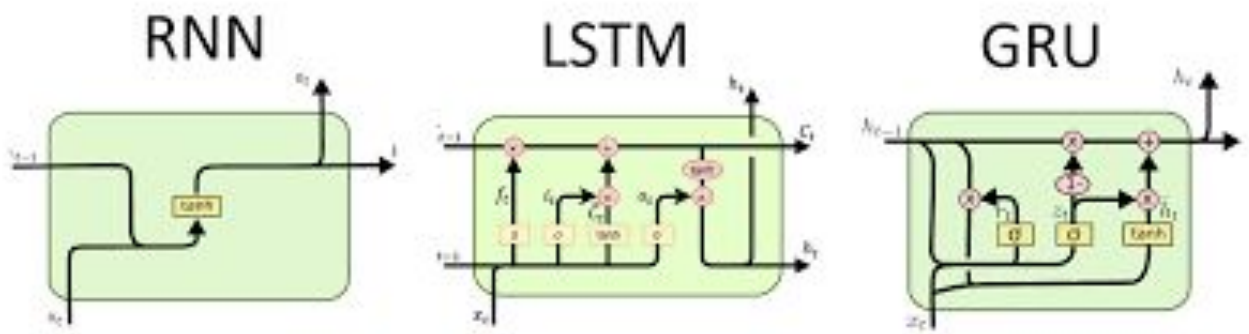
Backpropagation through Time (BPTT)

- Wir haben 3 Matrizen mit Gewichten: W, V, und U
- Die Loss Funktion muss nach jeder dieser Matrizen abgeleitet werden
- Wie das genau geht: https://jramapuram.github.io/ramblings/rnn-backrpop/



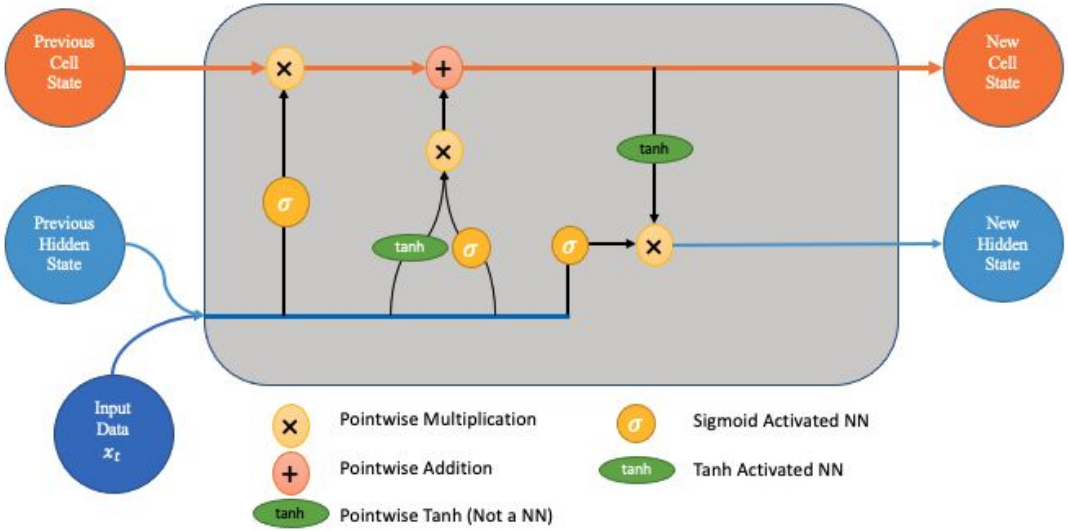
Schwindene/Explodierende Gradienten

- Beim Training mit BPTT müssen die Gradienten von der letzten Zelle bis zur ersten Zelle wandern.
- Das Produkt dieser Gradienten kann gegen Null gehen oder exponentiell ansteigen.
- Explodierenden Gradienten bezieht sich auf den starken Anstieg der Norm des Gradienten während des Trainings.
- Schwindenden Gradienten: wenn die Norm des Gradienten exponentiell schnell auf Null geht, was dem Modell unmöglich macht, eine Korrelation zwischen zeitlich entfernten Ereignissen zu lernen



LSTM (und Gated Recurrent Unit)

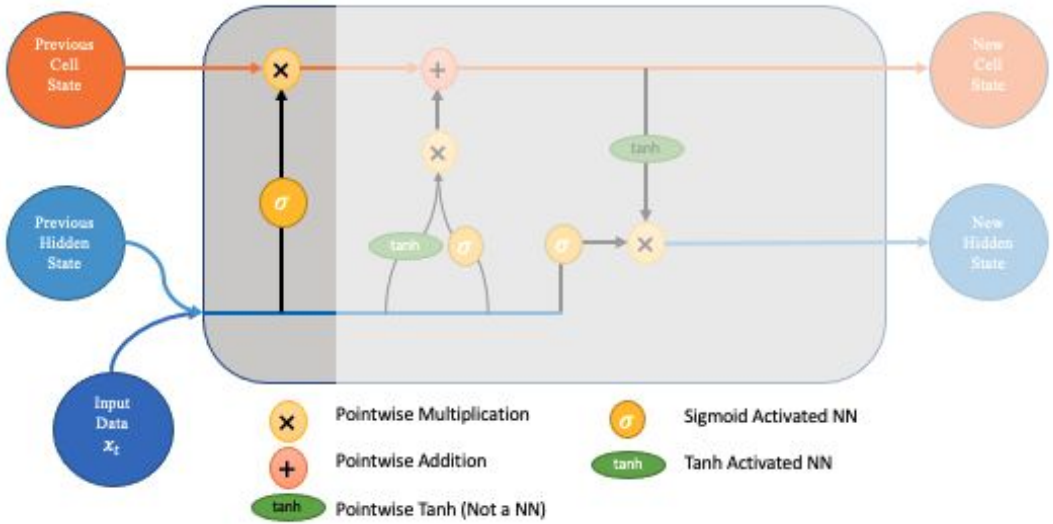
- Das Long-Short-Term-Memory Modell (erfunden von dem Österreicher: Sepp Hochreiter) löst das Problem der schwindenden Gradienten
- Durch das einführen von “Gates”



Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

LSTM 1)

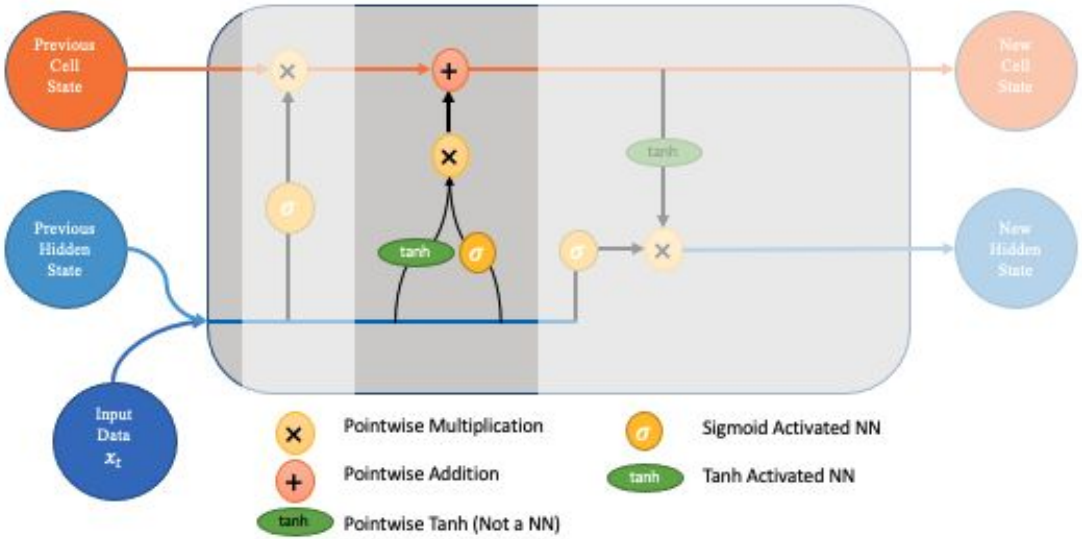
- Der Erste Schritt heißt “Forget Gate”:
- Hier werden wir entscheiden, welche Bits des Cell States (Langzeitspeicher des Netzwerks) angesichts sowohl des vorherigen Hidden State als auch neuer Inputs nützlich sind.



Source: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

LSTM 2)

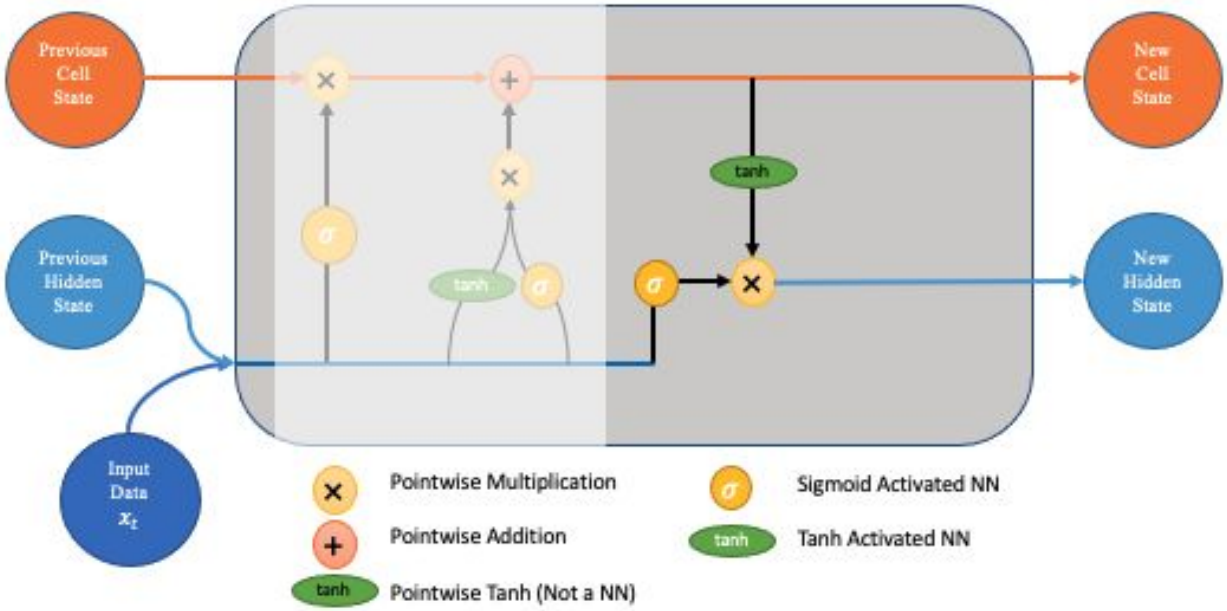
- Der zweite Schritt ist das New Memory Network und Input Gate.
- Das Ziel dieses Schritts besteht darin, zu bestimmen, welche neuen Informationen dem Langzeitgedächtnis (Zellzustand) des Netzwerks hinzugefügt werden sollten, wenn der vorherige verborgene Zustand und neue Eingabedaten gegeben sind.



Source: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

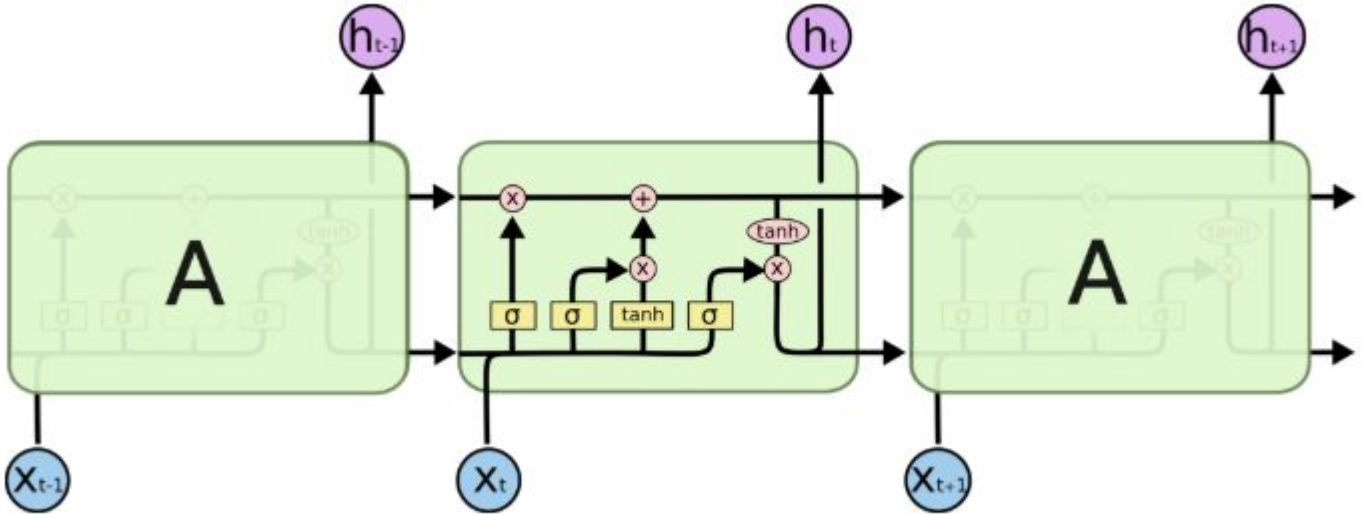
LSTM 3)

- Der zweite Schritt ist das Output Gate: um zu bestimmen was den neuen Hidden State betritt



Source: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

LSTM 4)

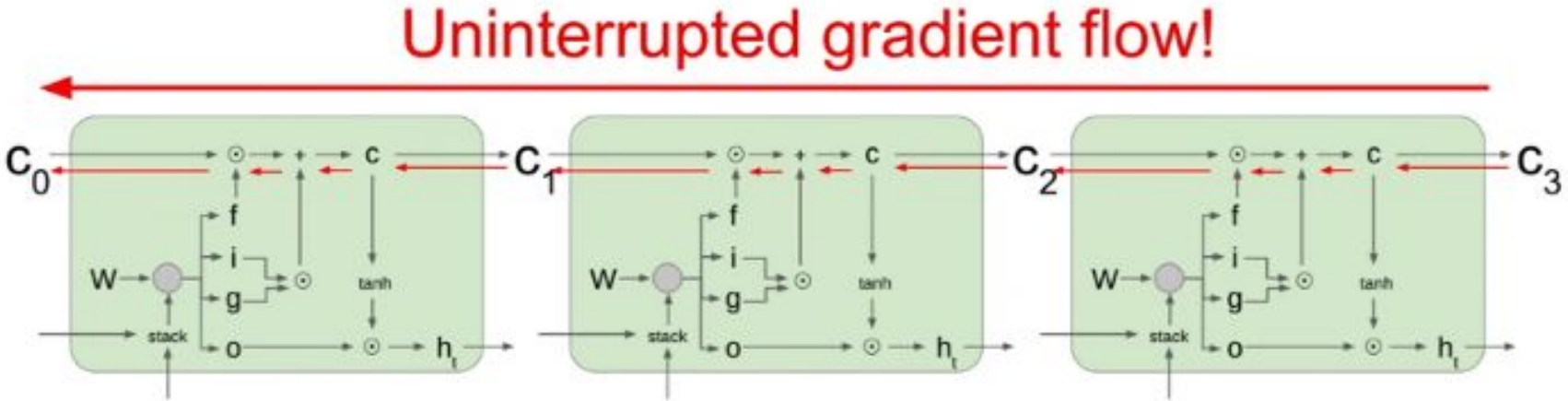


The repeating module in an LSTM contains four interacting layers.

Linear Layer

Source: https://medium.com/hackernoon/understanding-architecture-of-lstm-cell-from-scratch-with-code-8da40f0b71f4

LSTM 5): BPTT



Source: https://medium.com/hackernoon/understanding-architecture-of-lstm-cell-from-scratch-with-code-8da40f0b71f4

Notebook:

 Google Colab

1. Transfer Learning Demo

Ende

Fragen?

houben@eeg.tuwien.ac.at
+43 664 1545107
https://github.com/NikolausHouben/HAB_Strom