Dürfen wir auch mitspielen?

Deep Learning für Java-Entwickler

MATHEMA
CAMPUS

Motivation

Klassische Softwareentwicklung	**(konnektionistische) Künstliche Intelligenz**
logik-/ kontrollflussorientiert	daten-/ wahrscheinlichkeitsorientiert
Entwickler *programmiert* Programm, das abläuft und Lösung berechnet	Entwickler definiert ein neuronales Netz, das *trainiert* wird, um Lösungen zu finden
z. B. Parser, Compiler, State Machine	z. B. Perzeptron, Feed-Forward, CNN, RNN, ...
z. B. Backtracking, Rekursive Algos	z. B. Backpropagation, Transfer-Lernen
z. B. Quick Sort, Merge Sort, ...	z. B. genetische/evolutionäre Algorithmen
z. B. Tiefen-/Breitensuche/A*	z. B. Encoder/Decoder
Array, Liste, Stack, Baum, Graph, (Hash-)Tabelle, Key-Value-Paar ...	
Schichtenarchitektur, MVC, Event-Driven,	

Zitat

"Die Menschen, die zu meinen Vorträgen kommen, sollen mit Fragen heimgehen, nicht mit Rezepten."

Götz Werner
Unternehmer und Gründer von dm

Verschiedene Lernstufen

Offline

- überwacht (supervised learning)
- unüberwacht (unsupervised learning)

Online

- bestärkend (Reinforcement learning)

Anwendungen/Arbeitsmodi

- (Muster-/Daten-)Analyse
- Evaluation/Bewertung
- Diagnose
- Generierung
- Optimierung
- Projektion/Filterung
- Vorhersage/Prädiktion
- Dialog

Einsatzgebiete

- Verkehr & Logistik
- Finance
- Bildung
- Umweltschutz und Landwirtschaft
- Gesundheitswesen
- Rüstung und Cybersecurity
- Kunst und Kreativität
- Personalwesen und Soziales
- …

Los geht’s

- Entwicklungsumgebung aufsetzen
- Daten vorbereiten
- Netzwerk definieren
- Modell trainieren
- Modell tunen
- Modell in Produktion laden

Projekt-Setup

- https://deeplearning4j.konduit.ai/
- Code: 3 Kern-Dependencies
  - `deeplearning4j-core`: Definieren und Ausführen von Netzen
  - `nd4j-native`: numerical data (↑numpy), optimierter Code
  - `datavec-api`: zum Vektorisieren und Lesen von Daten
  - `deeplearning4j-ui`: Netzwerk- und Performance-Visualisierung
- Beispielprogramme → diese bitte klonen/herunterladen

  https://github.com/deeplearning4j/deeplearning4j-examples
- Dokumentation

  https://github.com/deeplearning4j/deeplearning4j-docs

Daten bereitstellen

- beschaffen (z. B. über https://kaggle.com/
- aufbereiten, säubern, projizieren, normalisieren
- organisieren (= aufteilen in)
  - training set
  - validation set
  - test set

Netzwerk bauen

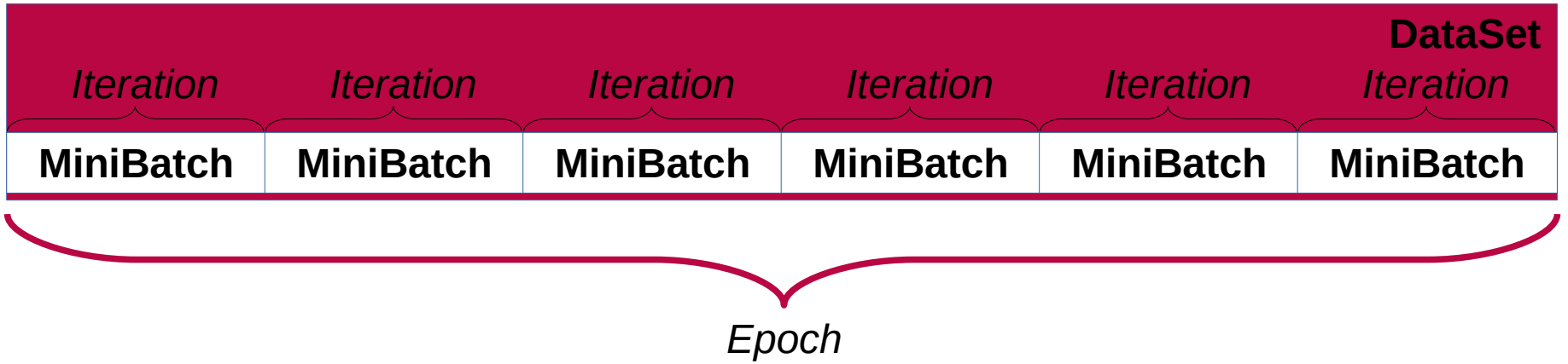
- Netzwerk-Topologie definieren

  Typ und Anzahl der Schichten (Layer)
- Netzwerkparameter einstellen
- Trainingsparameter einstellen

Training

- Ziel: ein trainiertes Modell
- das gut auf Training Set performt
- Fine Tuning der Hyperparameter auf Validation Set
- Gütebeurteilung auf Test Set
- alle drei Data Sets (training, validation, test) sind zwingend disjunkt
- zur Vermeidung von Überanpassung (Overfitting)

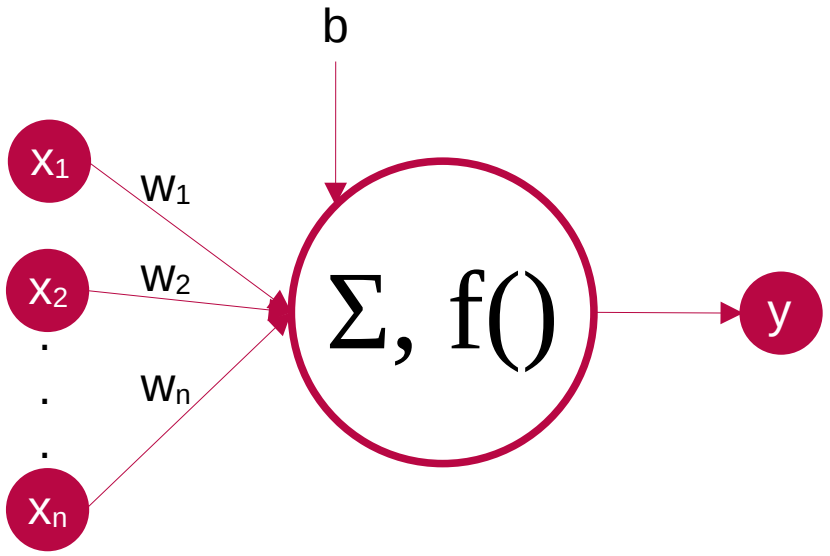
Training (2)



Beispiel

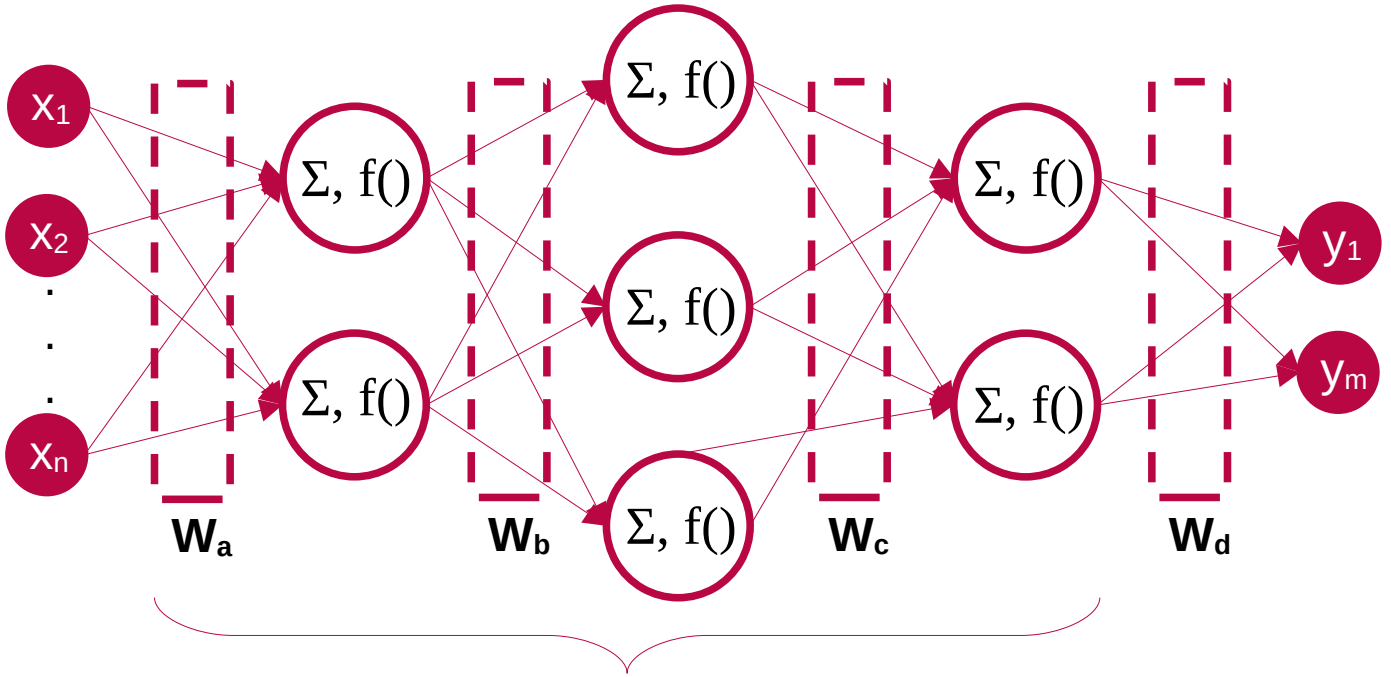
- Data Set Size = 200 Trainings Examples
- MiniBatch Size = 10
- → Anzahl an Iterationen (pro Epoche) = 20
- bei 5 Epochen → Netzwerk wird 1.000 Datensätze sehen

Das Neuron



Eingangssignale:	x_1 bis x_n
Gewichte:	b, w_1 bis w_n
Aktivierungsfunktion:	Sigmoid, Tanh, Heavyside, ReLU, ...
Ausgangssignal:	y

Feed Forward Network
MLP (Multilayer Perceptron)



Input-Layer

- keine Gewichte
- keine f()

Hidden-Layer

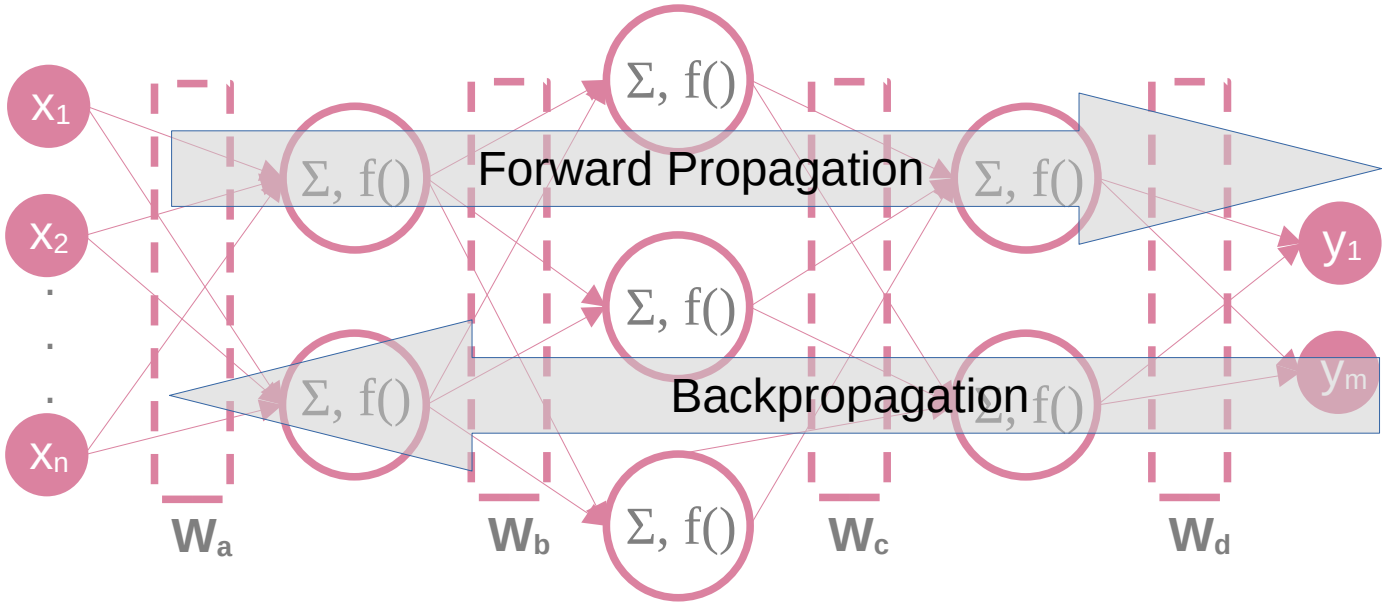
- Gewichte
- Aktivierungsfunktion f()

Output-Layer

- Gewichte
- f() optional

Feed Forward Network (2)

Training mit Backpropagation und Gradientenabstieg



- Datenpunkt am Input (X) anlegen und nach vorne propagieren (Forward Propagation)
- erwartetes Ergebnis (Label) am Output (Y) anlegen
- mittels Fehlerfunktion (Loss function, z. B. Mean-Square-Error) Fehler berechnen
- Fehler = Differenz zwischen IST-Output des Netzwerks und Label
- Fehler (um Lernrate und Momentum angepasst) zurückpropagieren und Gewichte anpassen

XOR: Hello, World!

- **DL4J-Repo**

  `org.deeplearning4j.examples.quickstart.modeling.feedforward.classification.ModelXOR`

- **Campus-Repo**

  `org.example.feedforward.ModelXOReasy`

Feed-Forward-Netz
supervised, Klassifikation

Wikipedia, by Suvanjanprasai

- **MNIST-Datensatz**
- Erkennung handschriftlicher Ziffern
- 60.000 Beispiele im Datensatz; Aufteilung z. B.
  - 48.000 als Training Set
  - 10.000 als Validation Set
  - 2.000 als Test Set

Trainingsaufwand vs. Performance

auf einer intel vPRO i7	**SingleLayer**	**DoubleLayer**
Trainingsdauer	1 min 50 sec	40 min
Modell-Performance (network score = Loss-function error)	0.094	0.008

geplant ist auch eine **Hardware-Optimierung**
für meine Architektur (`linux-x86_64-avx2`) habe ich diese jedoch nicht zum Bauen bekommen (`1.0.0-M2.1`).

vgl.:
https://community.konduit.ai/t/missing-artifacts-on-maven-central-in-nd4j-native-1-0-0-m2-1/1960/5

ND4J – N-Dimensional Arrarys for Java

- ähnlich Pythons NumPy
- effizientes Erzeugen und Transformieren von n-dimensionalen Arrays (Vektoren, Matrizzen, Tensoren)
- auf unterschiedlichen Plattformen: CPU, GPU, TPU
- Normalisieren von Werten (0..1), z.B. ImagePreProcessingScaler
- ```
  INDArray vector = Nd4j.create(new float[]{5, 6});
  INDArray matrix = Nd4j.create(new float[][]{{2, 3}, {4, 5}});
  INDArray tensor = Nd4j.rand(new int[]{2, 3, 4});
  ```
- ```
  matrix.transpose();
  INDArray result = matrix.mmul(vector);
  ```

DataVec – Vorverarbeitung (ETL)

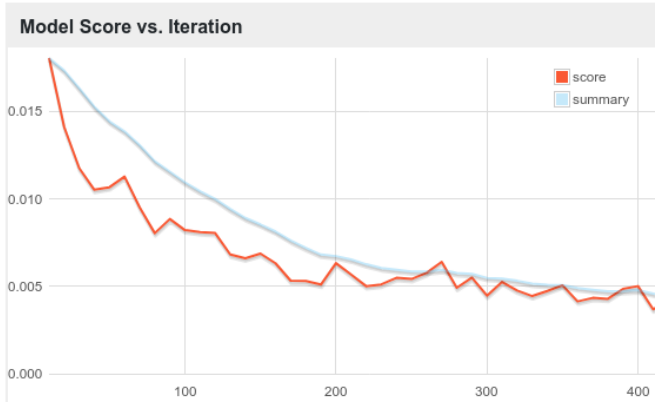
- Build-in-Transformationen zum Konvertieren
- Input: kann text, csv, image, video
- Output: SVMLight-Format

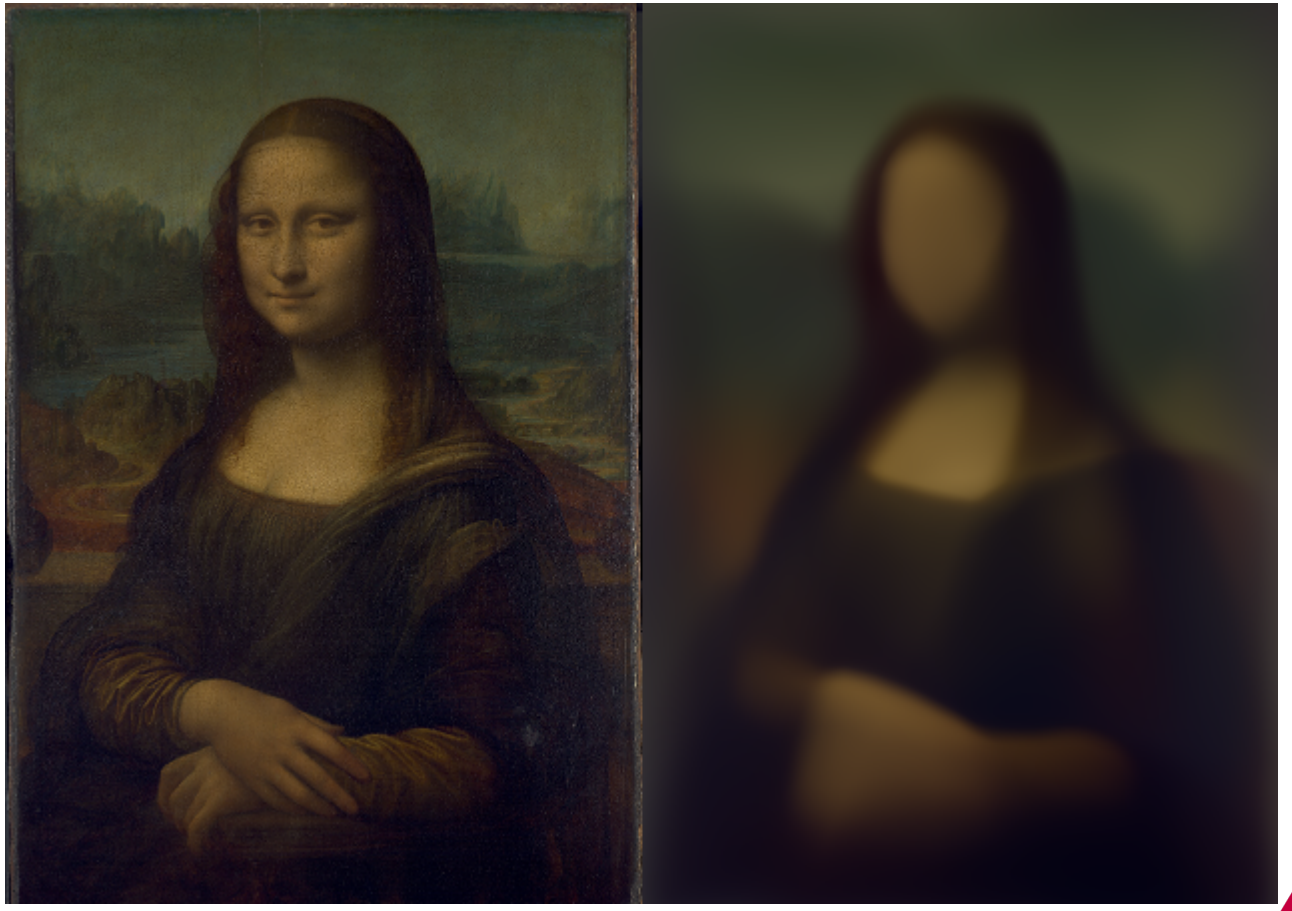
Aufgabe – Feed-Forward-Netz

- Trainiere ein Modell auf dem MNIST-Datensatz
- exportiere das Modell in eine Datei
- lade in einem eigenen Programm die trainierte Modelldatei und führe es aus, sodass Ziffern aus einem eigenen Datensatz klassifiziert werden

Regression, unsupervised– Feed-Forward-Netz

- ImageDrawer





Convolutional Neural Network (CNN) – supervised, Klassifikation

- was, wenn unsere Ziffern verschoben sind?
- oder gedreht? → Translations-/Dreh-Invarianz?
- oder unsere Bilder größer als 32x32 Pixel?

  → „Kombinatorische Explosion“
- Lösung: wir schalten Faltungs- (Convolution), Subsampling- und Pooling-Schichten vor unser Netzwerk
  - Faltungsschicht schafft Invarianz durch automatische Feature-Selektion
  - Subsampling-Schicht verringert die Anzahl der Gewichte

Faltung: Was war das nochmal?

$$f(x,y) \bullet g(x,y) = \sum_{n=0}^{\text{cols}} \sum_{m=0}^{\text{rows}} g(n,m) * f(x-n, y-m)$$

- Faltung zweier Matrizzen: Bild (image) f und Kern (kernel) g
- Es gibt viele Kernel (Merkmalsextraktoren), z. B. Kantendetektoren oder Schärfen-Filter
- auf die konkreten kommt es nicht an, sondern die CNN-Schicht berechnet „alle/viele“ und selektiert im Training diejenigen (Pooling) mit dem geringsten Fehler

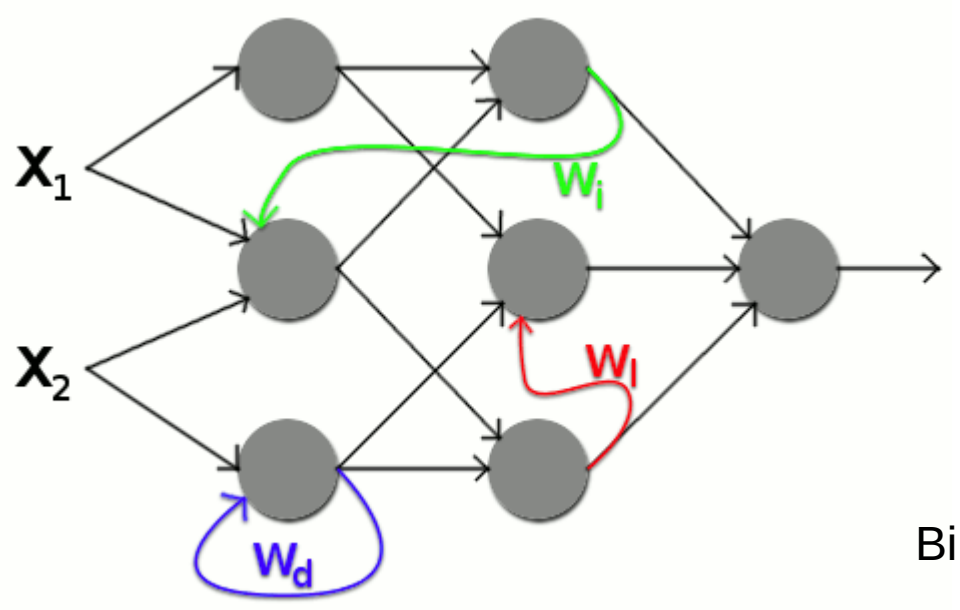
Recurrent Neural Network (RNN) – supervised, zeitliche Regression

- Google Translator (NLP natural language processing)
- Zeitreihenprädiktion (z.B. Börsenkurse, Temperaturwerte, Beschäftigungszahlen, …)
- Lernen von Zusammenhängen zwischen weit entfernt liegenden Datenpunkten
- Korrelationen zwischen Trainingsbeispielen
- BPTT (Backpropagation Through Time)

RNN (2)

- Findet Korrelationen auch zwischen weit entfernt liegenden Merkmalen
- z. B. gerade im Deutschen können bedeutungstragende Worte weit außeinander stehen
- es werden zum ersten Mal Zyklen (Rekurrenten) in der Netzwerk-Topologie eingeführt (zwischen Neuronen oder ganzen Schichten)
- Das Netz lernt somit Korrelationen von „alten“ zu neueren Werten

RNN (3)



Bildquelle: Wikipedia, by Mercyse

- die grüne Verbindung ist eine zeitliche Rückkopplung
- Das Problem der Vanishing/Exploding Gradients hat zu neuen neuronalen Architekturen geführt
  - LSTM (long short-term memory)
  - GRU (gated recurrent unit)

Literatur/Quellen

- Mitchell, Tom: **Machine Learning**, McGraw Hill, 1997.
- Russell, Stuart/Norvig, Peter: **Artificial Intelligence**: *A Modern Approach*, Pearson, 2021.
- https://deeplearning4j.konduit.ai/
- Udemy-Kurs: Artificial Intelligence III - Deep Learning in Java

MATHEMA
CAMPUS

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT

Email: consulting@thomas-a-bertz.de

thomas bertz
{ development | consulting | training }
it-consulting