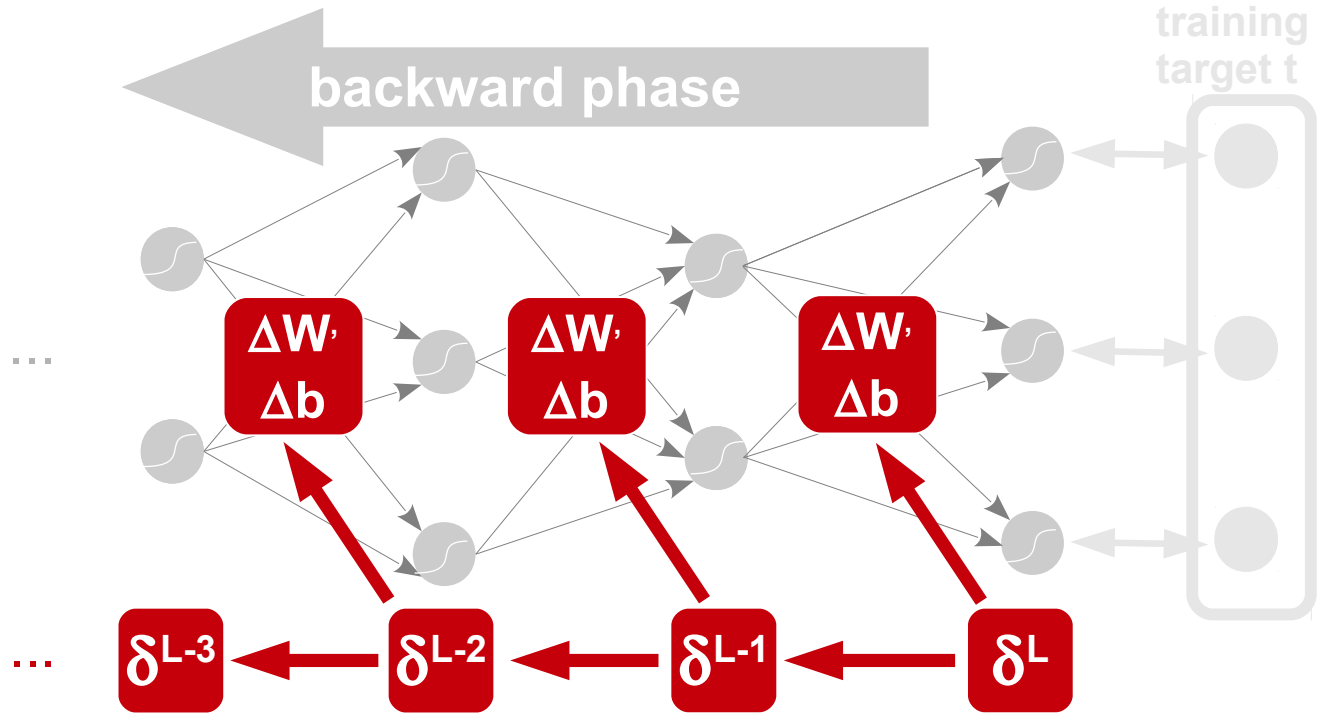
Anwendungen der KI

– Sommersemester 2018 –

Kapitel 07: Neuronale Netze II

Prof. Dr. Adrian Ulges
B.Sc. Informatik (AI, ITS, MI, WI)
Fachbereich DCSM
Hochschule RheinMain

Backpropagation (Wdhlg.)



Backprop-Formeln

$$\delta^L = \left(\mathbf{a}^L - \mathbf{t}\right) \odot f'(\mathbf{z}^L)$$

$$\delta^l = \left(W^{l+1} \cdot \delta^{l+1}\right) \odot f'(\mathbf{z}^l)$$

$$\Delta w_{ij}^l = -\lambda \cdot \delta_j^l \cdot a_i^{l-1}$$

$$\Delta b_j^l = -\lambda \cdot \delta_j^l$$

Outline

Backpropagation: Batching Bild: [2]

Stochastischer Gradientenabstieg

Wähle in jeder Iteration ein Trainings-Sample k und berechne Gewichts-Updates $w_{ij}^l(k)$ für dieses Sample.

'Eigentlicher' Gradientenabstieg

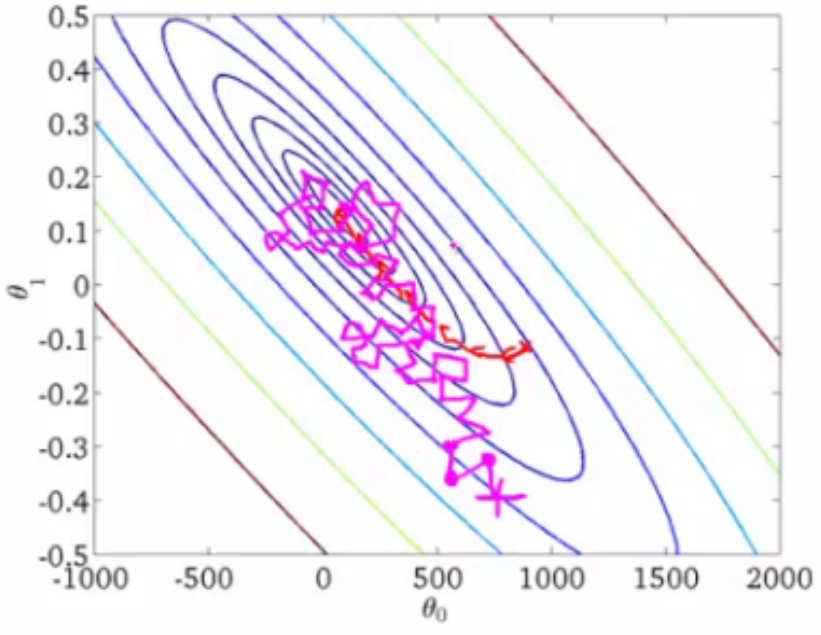
In der Theorie wollen wir den Loss über **alle** Samples minimieren:

1. Berechne für alle n Trainingssamples die Updates, d.h. $\Delta w_{ij}^l(1), \Delta w_{ij}^l(2), ..., \Delta w_{ij}^l(n)$.
2. Das Gewichts-Update wird gemittelt: $\Delta w_{ij}^l := \frac{1}{n} \sum_k \Delta w_{ij}^l(k)$

Unterschied?

Schritte im 'normalen' Gradientenabstieg sind...

- glatter *(→ größeres λ)*
- deutlich teurer
- parallelisierbar!



Backpropagation: Minibatching

Üblicher Kompromiss

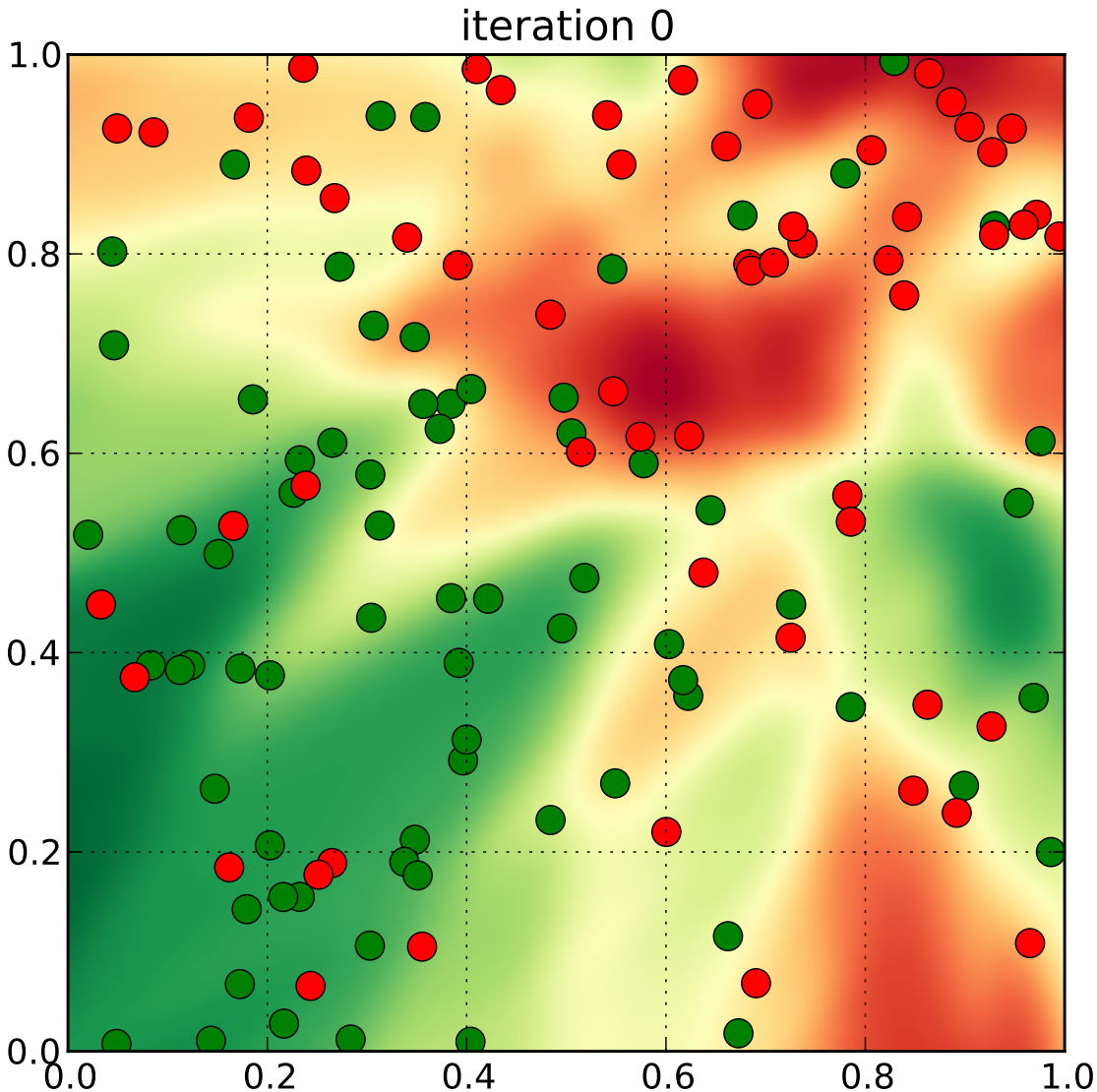
- Wir samplen in jeder Iteration eine **Submenge** von Samples, die sogenannte **Minibatch**
- Wir berechnen die Gewichts-Updates durch **Mittlung** über alle Samples in der **Minibatch**
- Wir trainieren ein paar Iterationen und wechseln dann zur **nächsten Minibatch**.

Optimale Minibatch-Größe?

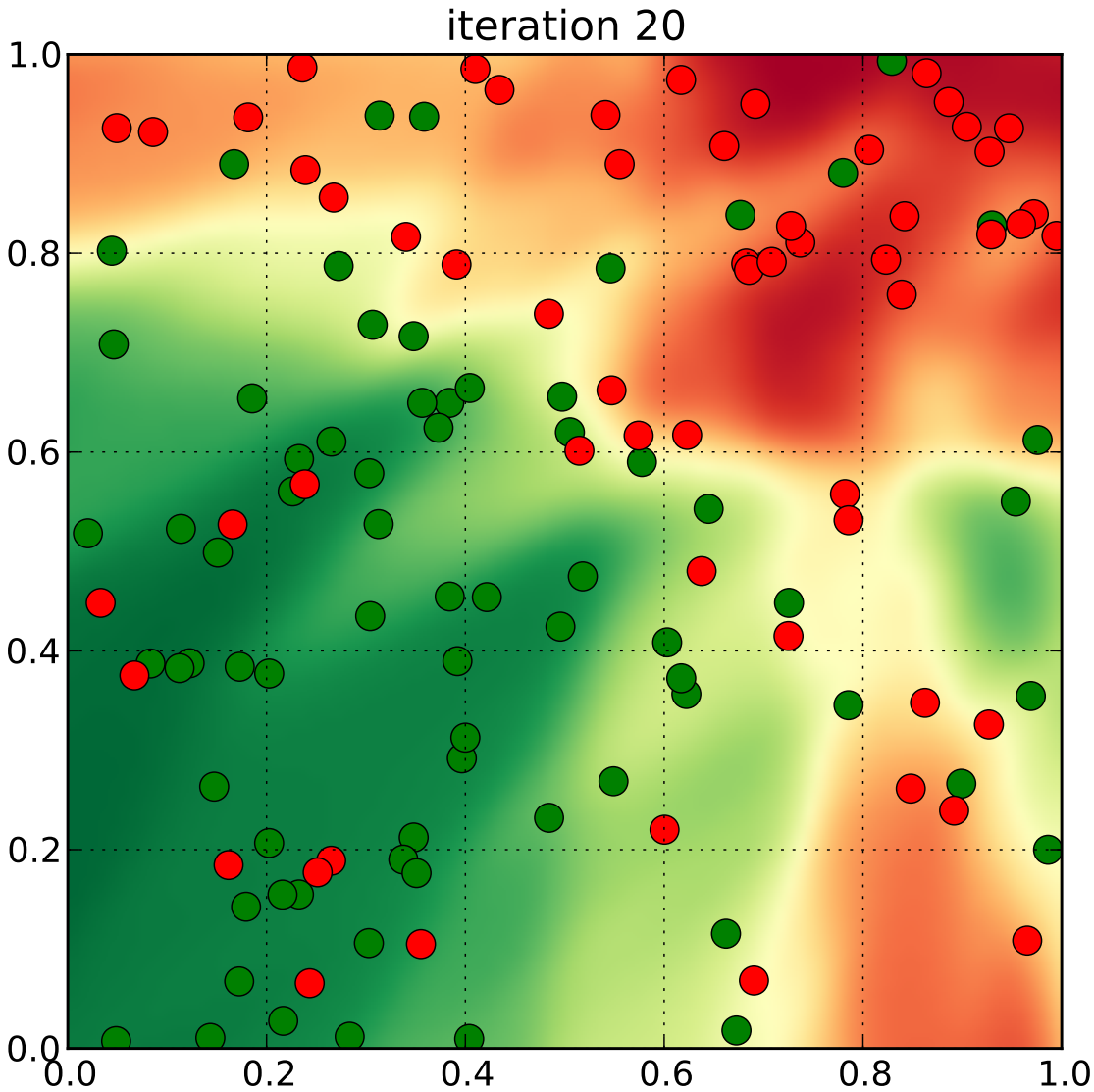
- **Üblich bei GPUs**: Wähle die Batch-Größe so, dass die Batch (gerade so) in den Speicher der (GPU) passt → Maximale Ausnutzung der Parallelisierung

> *"To set the minibatch size, plot the validation accuracy versus time (as in, real elapsed time, not epoch!), and choose whichever minibatch size gives you the most rapid improvement in accuracy."*
>
> (Nielsen: "Neural Networks and Deep Learning")

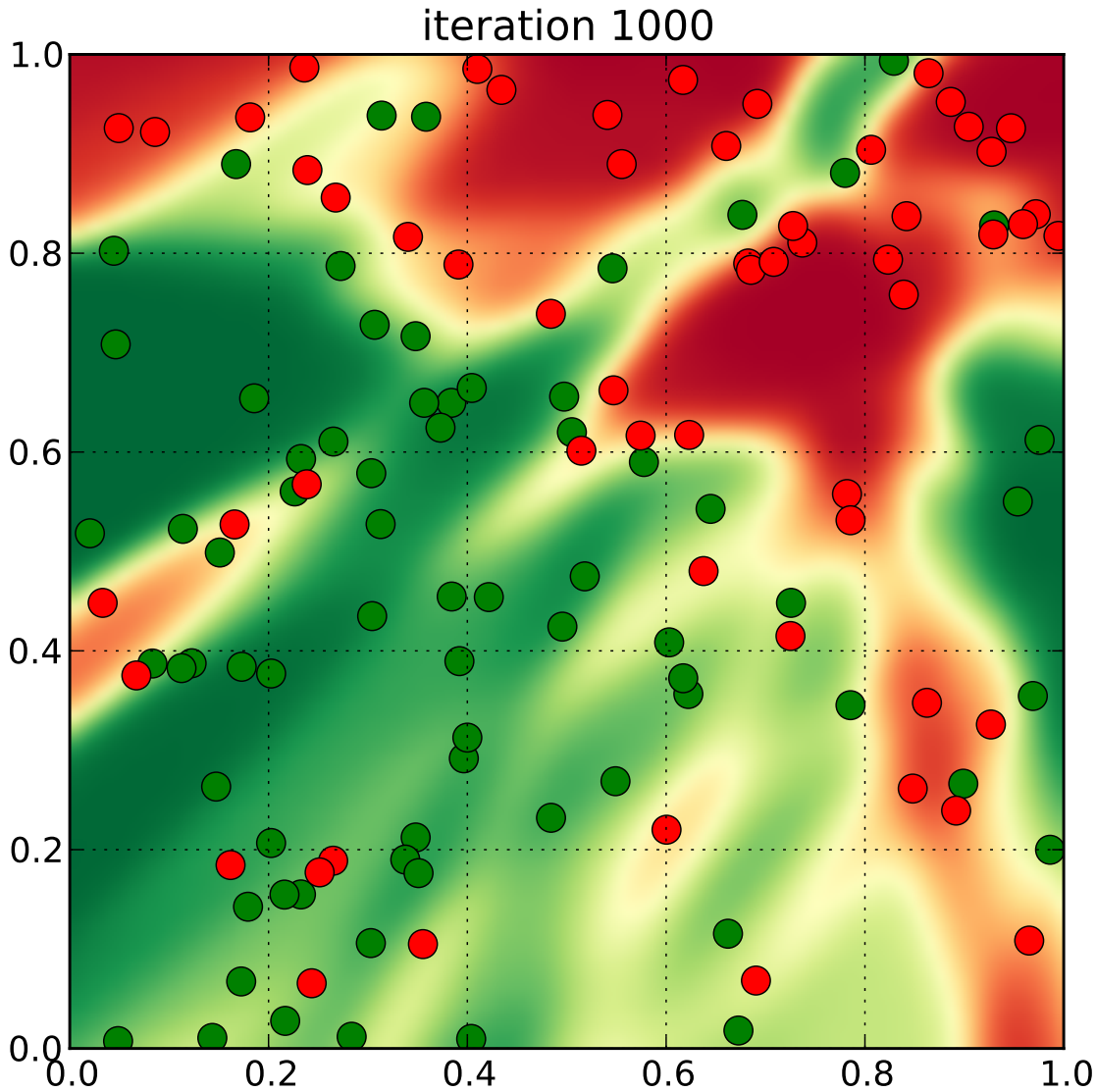
MLP: Beispiel-Training



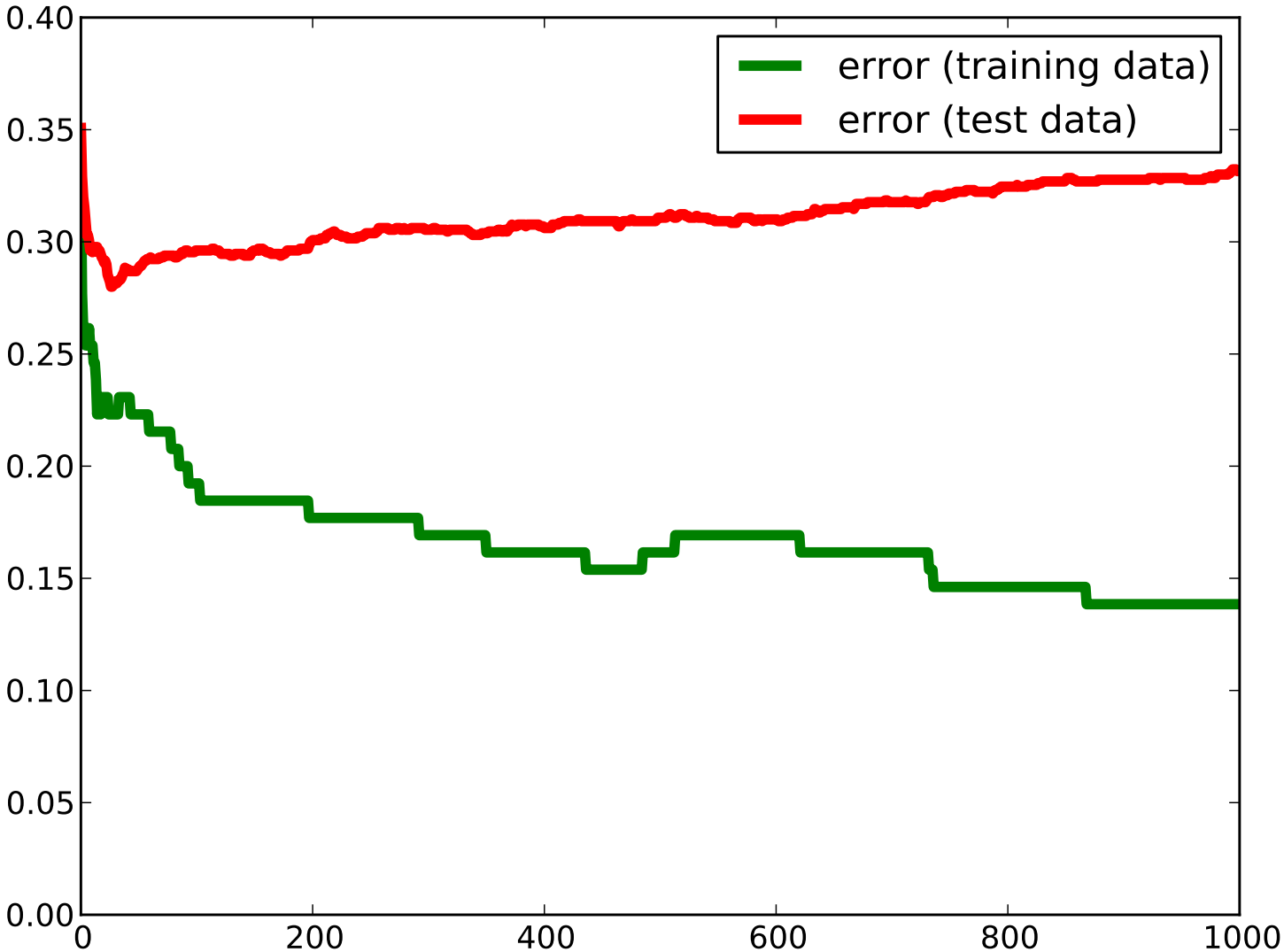
MLP: Beispiel-Training



MLP: Beispiel-Training



MLP: Beispiel-Training



Wann sollten wir Backpropagation abbrechen?

Üblich: “Early Stopping”

- Die **Anzahl der Lerniterationen vor** dem Training zu wählen ist schwer (sie hängt z.B. vom Startpunkt des Trainings, d.h. der Zufallsinitialisierung der Gewichte ab)
- Stattdessen zeichnen wir *während* des Trainings die Fehlerrate auf einer separaten **Validierungsmenge** auf
- Wir brechen ab, wenn sich diese Fehlerrate beginnt zu **verschlechtern**

Lost in Hyperparameter Space

Neuronale Netze haben sehr viele freie Parameter! Welche?

1. Netzwerk-Topologie
   - Anzahl an Schichten sowie Knoten per Schicht
   - Verbindungen zwischen Schichten
   - Aktivierungsfunktionen
2. Training
   - Lernrate, Anzahl der Iterationen, Minibatch-Größe
   - Initialisierung der Gewichte und Biases
   - Loss-Funktion
3. Weiteres
   - Eingabedaten: Vorverarbeitung (Normalisierung, Balancierung)
   - ...

Lost in Hyperparameter Space

> *"When you understand something poorly - as the explorers understood geography, and as we understand neural nets today - it's more important to explore boldly than it is to be rigorously correct in every step of your thinking."*
>
> (M. Nielsen)

Wichtiger Tip

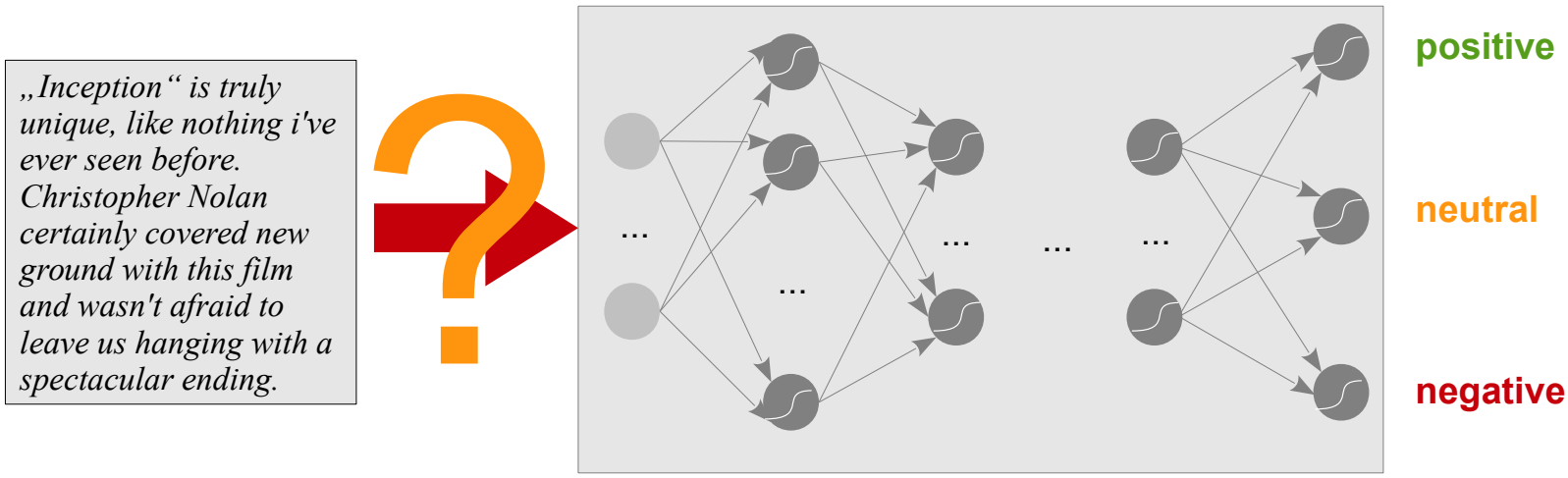
- **Schnell explorieren!** → Iteriere über kleine Teile der Daten, um vielversprechende Konfigurationen zu finden!

Literatur

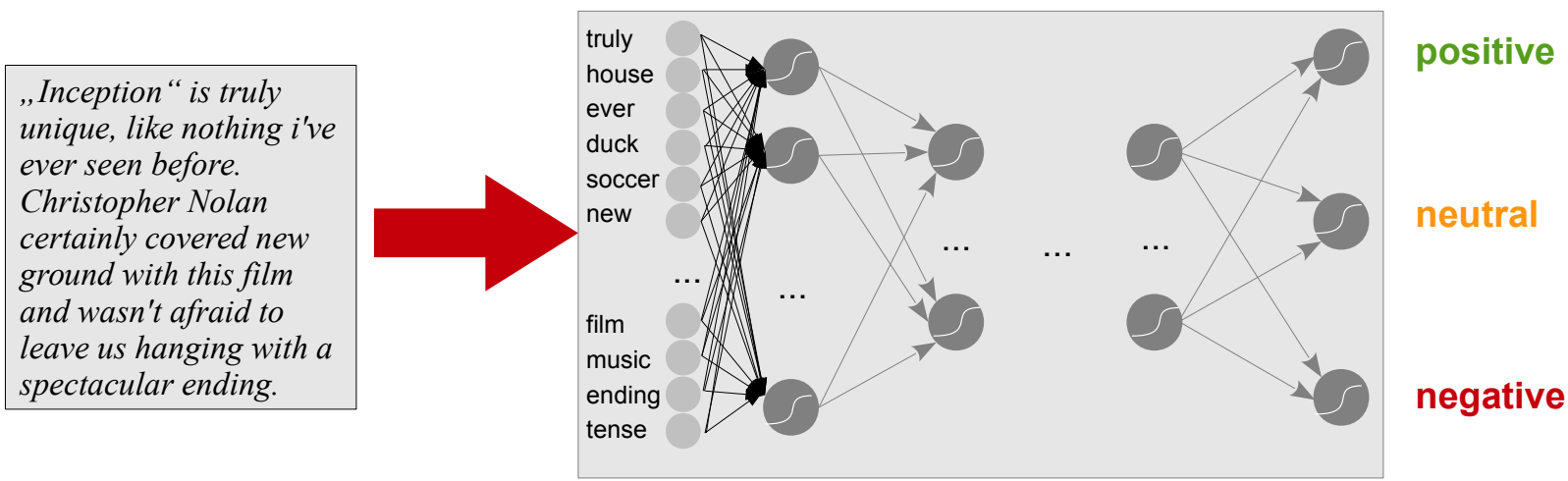
- Bengio: "Practical recommendations for gradient-based training of deep architectures" (2012).
- Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller: "Neural Networks: Tricks of the Trade" (2012).

Outline

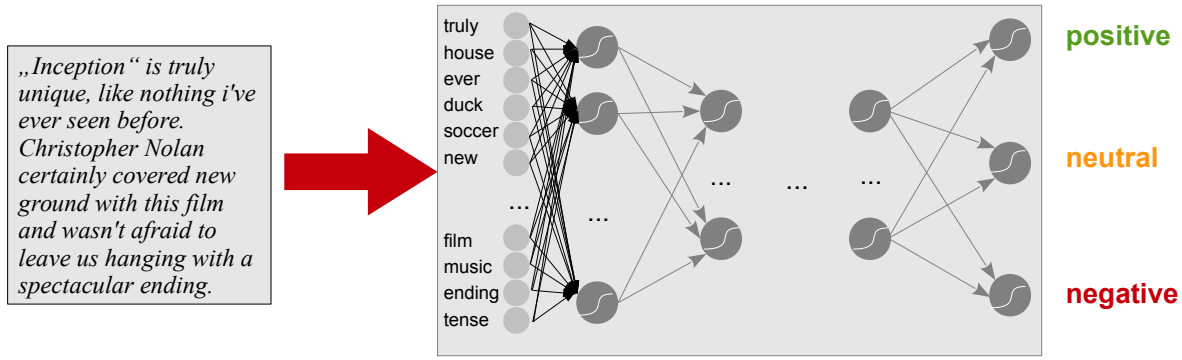
Neuronale Netze im NLP



- Neuronale Netze benötigen **numerische Eingabevektoren**. Wie gewinnen wir diese aus Text?
- Bisherige Ansätze: z.B. **Bag-of-Words**



Limitierungen bisheriger Techniken (Bag-of-Words)



- Größe der Eingabe = #Terme = **riesig** (z.B. 100K)
  (obwohl Eingabetexte oft **sehr kurz** *sind)*
- Die **Reihenfolge** der Terme wird nicht berücksichtigt:

  “It was not good, it was actually quite bad” vs.
  “It was not bad, it was actually quite good”

- N-Gramme ☺ ? → Noch mehr Dimensionen ☹
- Semantisch **ähnliche Worte** werden nicht berücksichtigt:

  “Trump speaks to the media in Illinois” vs.
  “The President greets the press in Chicago.”

- Terme, die in unseren (begrenzten) **Trainingsdaten nicht vorkommen**, können nicht berücksichtigt werden.

Limitierungen bisheriger Techniken (Lineare Modelle)

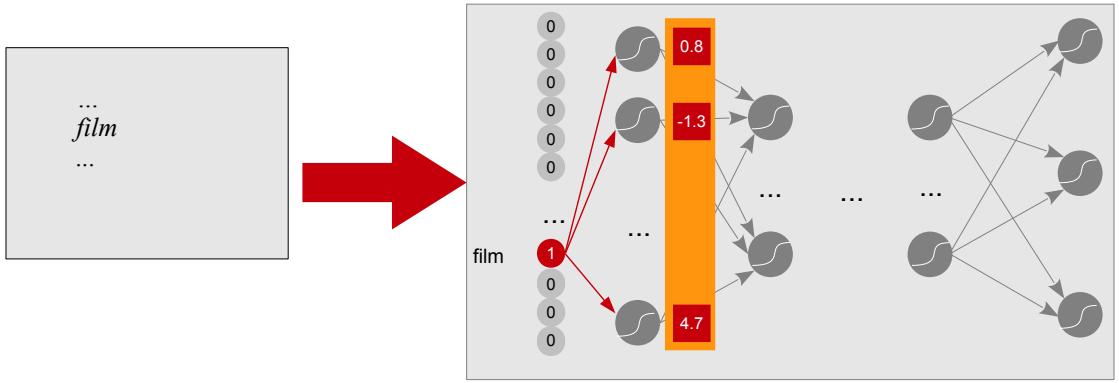
- **Lineares** Beispiel-Modell: **Logistische Regression**
- Lineare Modelle beurteilen die Eingabemerkmale $x_1, ..., x_n$ **unabhängig** voneinander (jedes erhält ein **Gewicht**)
- Ist das für NLP-Probleme **ausreichend**?
  Nein: Es existieren **Abhängigkeiten** zwischen Merkmalen!

Beispiel

> *"The* **bear** *survives in summer on fish and fruit." vs.*
> *"Your efforts will* **bear** *fruit."*

- Nur Satz 1 ist der Kategorie "Wildlife" zuzurechnen. Warum? Weil sowohl das Merkmal `bear (TERM)` als auch das Merkmal `Noun (POS)` aktiviert sind.
- Es gibt einen **Querbezug** zwischen beiden Merkmalen!
- Traditioneller Ansatz: Führe **zusätzliche Merkmale** ein, wie z.B. `bear(TERM)` $\wedge$ `Noun(POS)` *(manuelles Engineering)*

Neuronale Netze im NLP: Embeddings

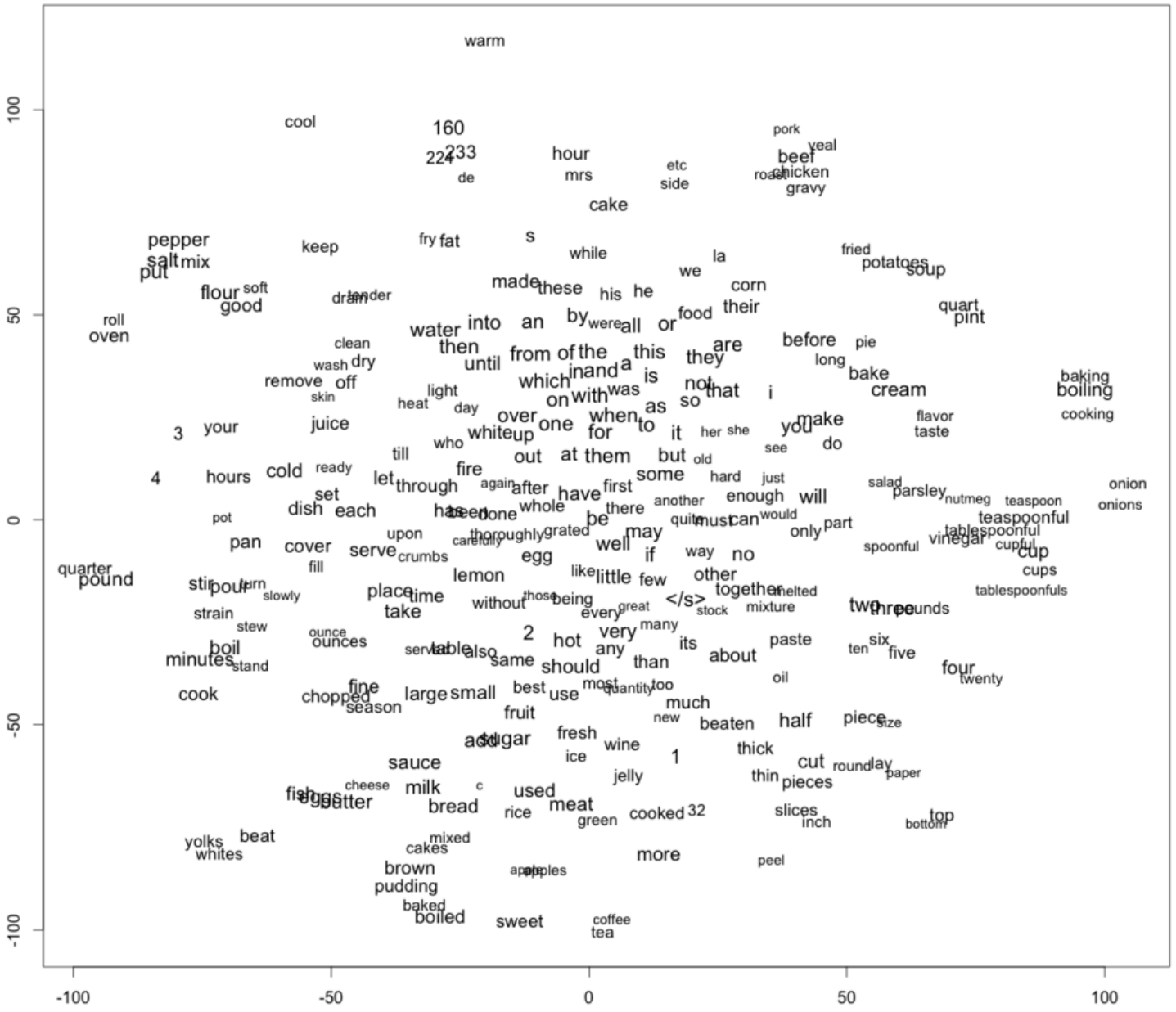


- Gedankenspiel: **One-Hot-Encoding** der Eingabe
- Die Aktivierungen der m Neuronen der ersten Hidden-Schicht formen einen **m-dimensionalen Vektor**!
- Diesen Schritt bezeichnet man als **Embedding e** *(Terme werden in einen m-dimensionalen Raum "eingebettet")*:

$$e : V \to \mathbb{R}^m, \text{ z.B. } \quad e(''film'') = (0.8, -1.3, ..., 4.7)$$

- Wir können die erste Schicht somit als eine **Lookup-Tabelle** auffassen, oder als eine Embedding-Matrix $E \in \mathbb{R}^{\#V \times m}$ *(E enthält für jeden Term eine Zeile)*

Die Embedding-Matrix E: Beispiel



Die Embedding-Matrix E: Anmerkungen

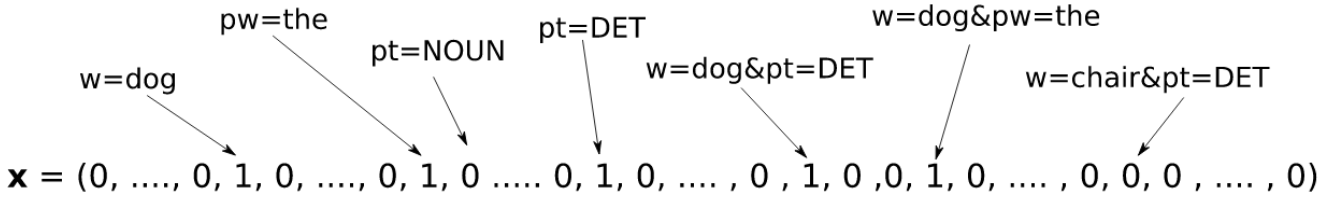
- E sollte für ähnliche Terme ähnliche Embeddings liefern!
  Warum? → **Generalisierung**, z.B. “Trump” ≈ “president”
- Wie berechnen wir die Term-Embeddings?

Ansatz hier: Unüberwachtes Training

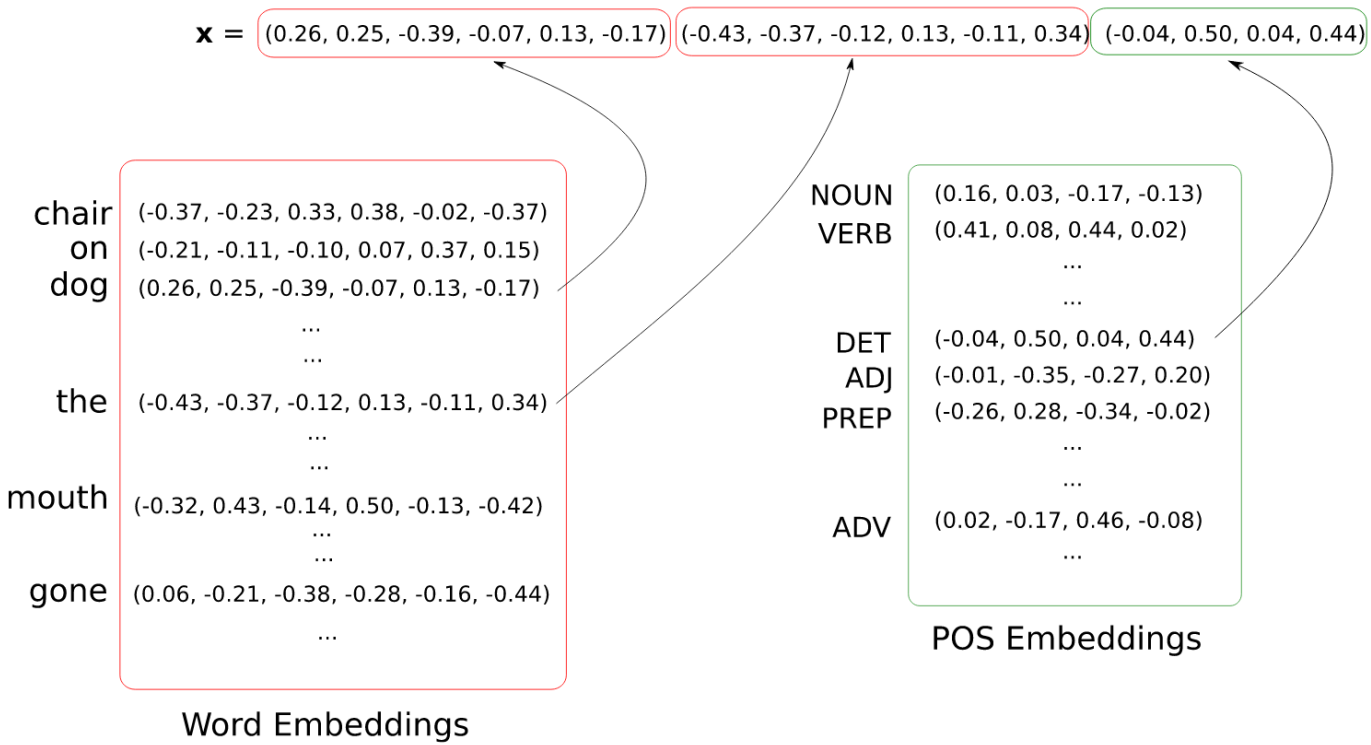
- Verwendung Neuronaler Netze als **Sprachmodelle** *(später)*
- **Keine Labels**, aber sehr großer Korpus *(1 Mrd. Terme)*
- Kombination beider Ansätze möglich!

Vergleich von NLP-Modellen Bilder: [1]

“Alt”: Darstellung dünn besetzt (“sparse”)



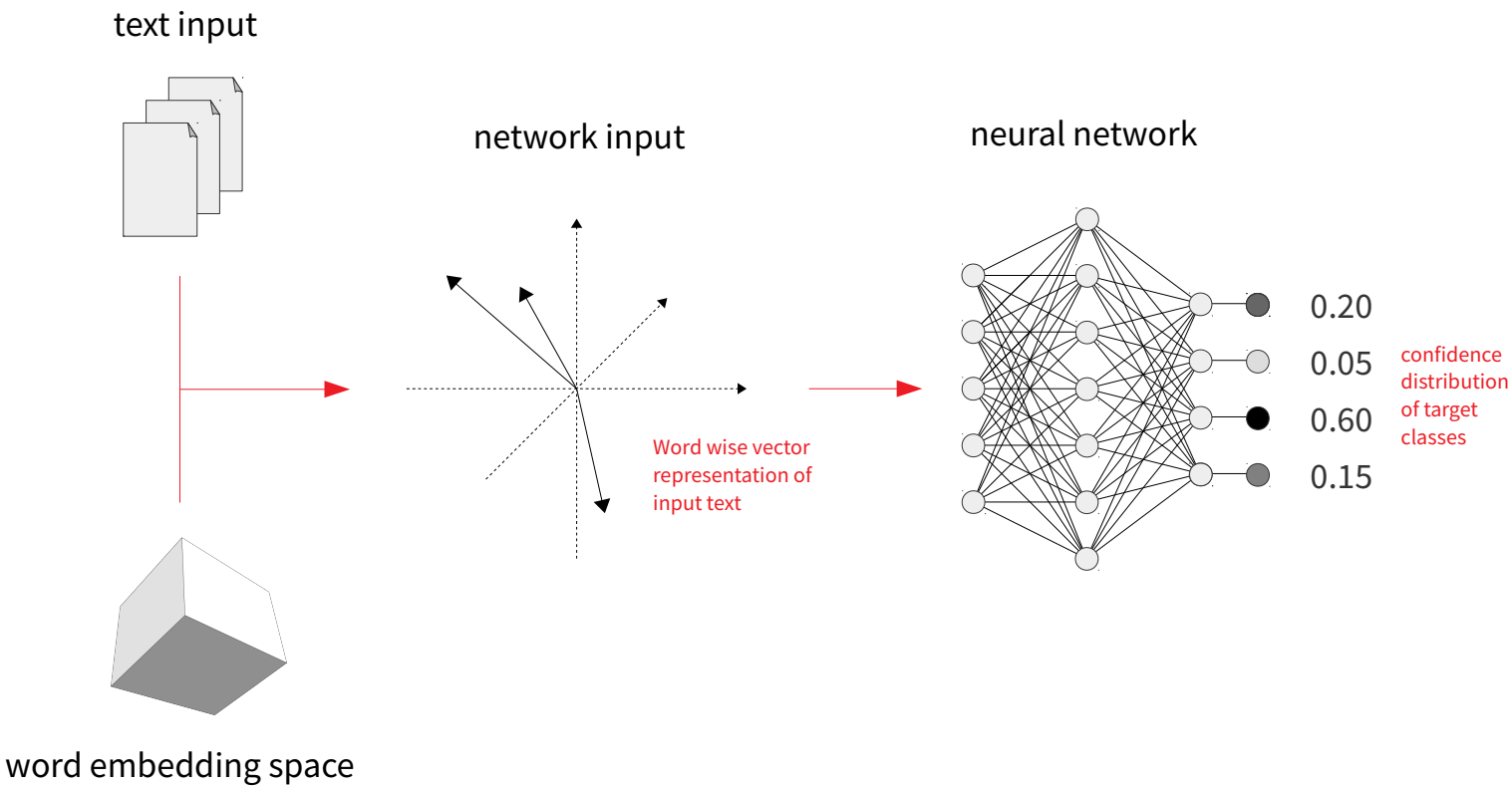
“Neu”: Darstellung dicht besetzt (“dense”)



Highlevel Overview

Generelles Vorgehen im Folgenden

- Text-Embeddings als Eingabe für neuronale Netze
- Neuronales Netz (das *"Downstream Model"*) löst z.B. ein Klassifikationsproblem



Outline

MLPs mit Embeddings Bild: [1]

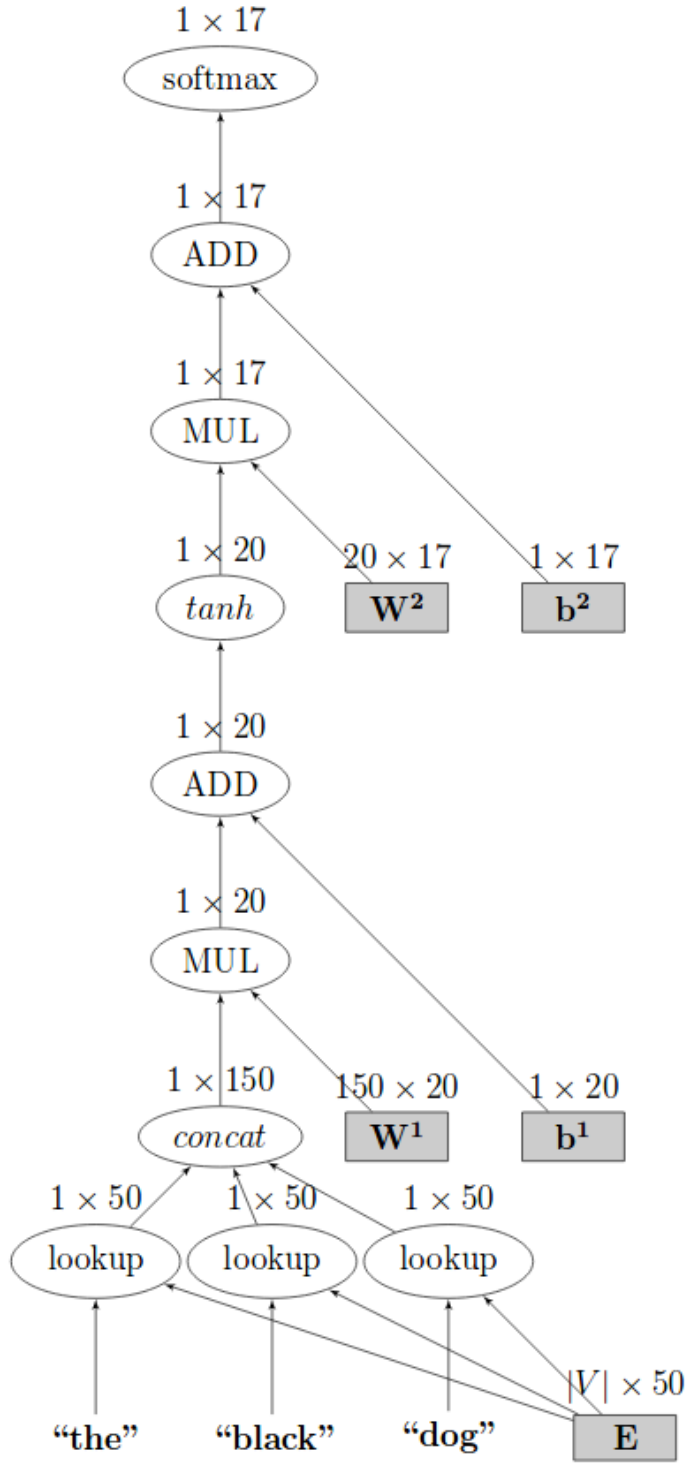
- Aufgabe: Klassifiziere einen Satz / ein Dokument
- Idee: Verwende die **Embedding-Vektoren** als Input für ein MLP
- Durch **weitere Schichten** erhalten wir höhere Ausdrucksmächtigkeit

Vorverarbeitung

- Für Terme $t_1, ..., t_n$ erhalten wir Embeddings $e(t_1), ..., e(t_n) \in \mathbb{R}^m$
- Vor Eingabe in das Netz **kombinieren** wir die Embeddings, z.B. durch **Mittlung** *("continuous bag of words")* oder **Konkatenation**:

$$\mathbf{x} = \frac{1}{n} \sum_i e(t_i) \quad \text{oder} \quad \mathbf{x} = e(t_1) \circ ... \circ e(t_n) \in \mathbb{R}^{n \cdot m}$$

- Achtung: Im Falle der Konkatenation muss **x** für Dokumente/Sätze unterschiedlicher Länge gleich lang sein. Deshalb **füllen** wir bis zu einer **Maximallänge** mit Nullen **auf** *(engl. "padding")*



References I

[1] Goldberg, Yoav.
A Primer on Neural Network Models for Natural Language Processing.
http://u.cs.biu.ac.il/~yogo/nnlp.pdf, retrieved: Apr 2017).

[2] A. Holehouse.
Stanford Machine Learning (Transcript of Course by Prof. Andrew Ng).
http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html (retrieved: Nov 2016).