

5.5) Aktivierungsfunktionen im Vergleich

Beim Setup eines Neuronalen Netzes spielt die Aktivierungsfunktion eine entscheidende Rolle ...

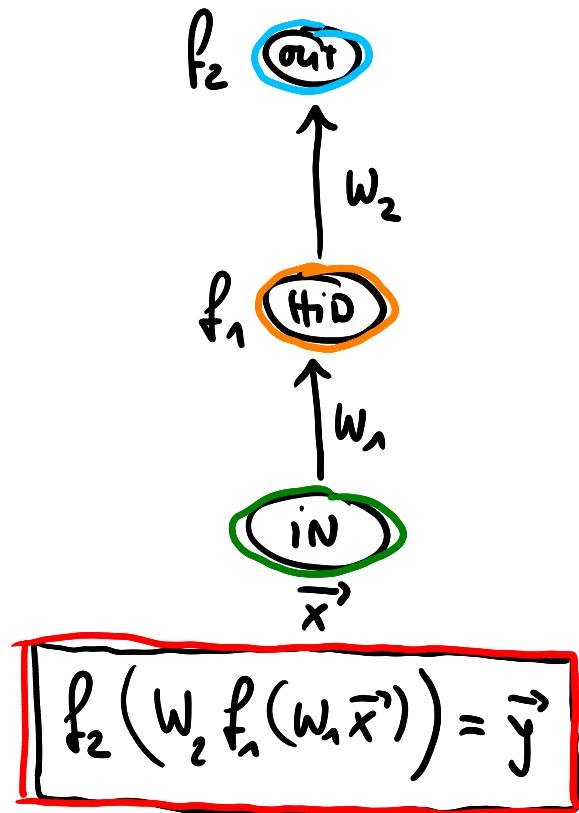
Beim Vorwärtslauf:

→ klar, da f_1, f_2 direkt die Transferfunktion mit bestimmen

Aber auch sehr wichtig im Rückwärtspfad:

➡ GRADIENTENABSTIEG

BACKPROPAGATION



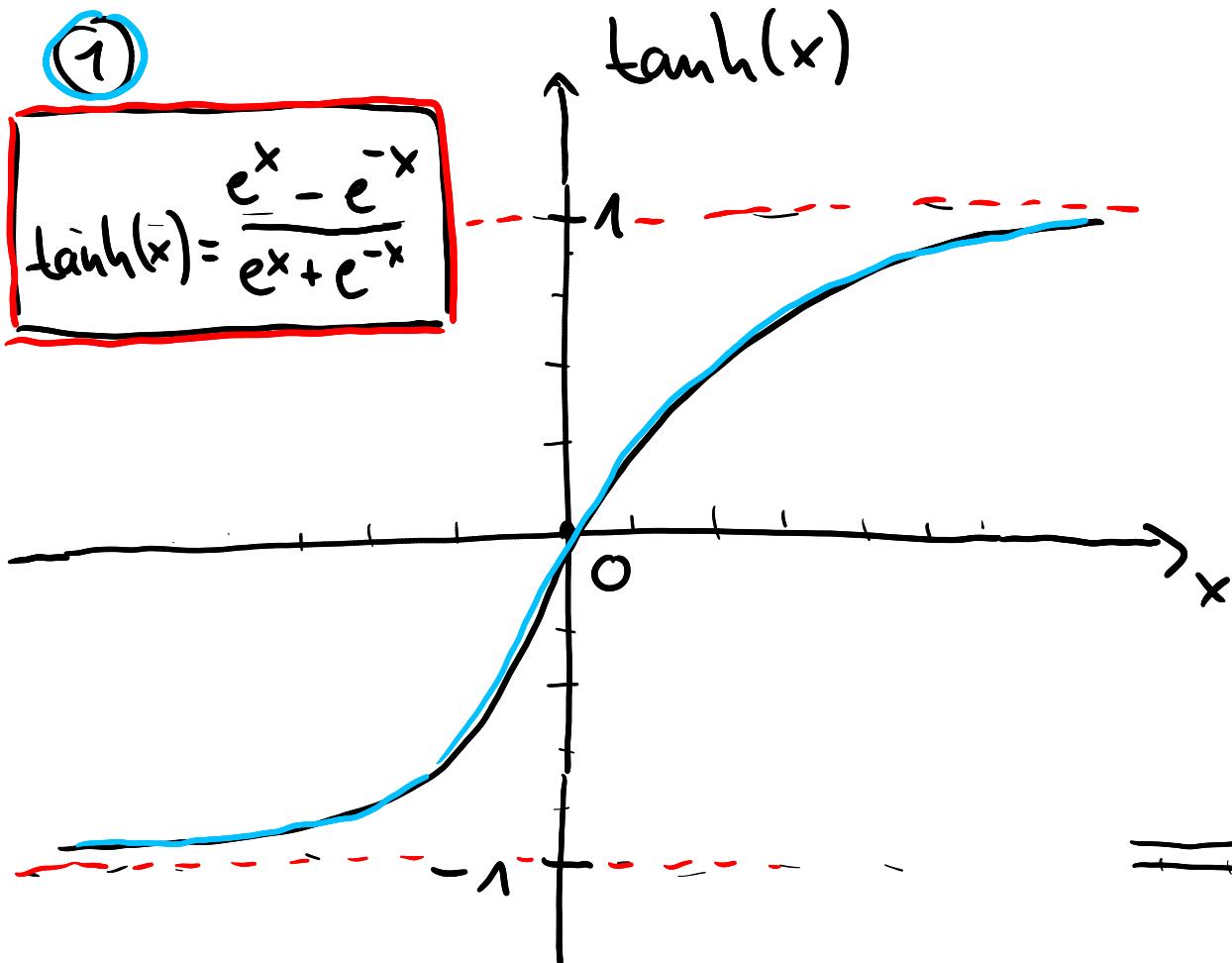
Zwei Beispiele:

330

①

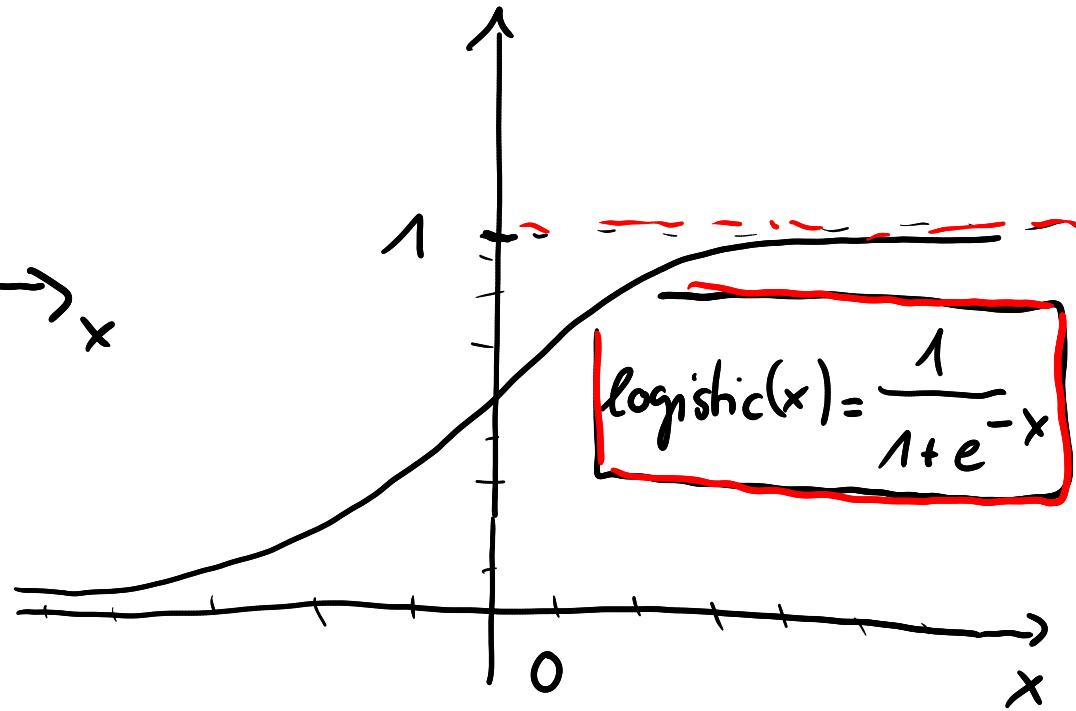
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$\tanh(x)$



②

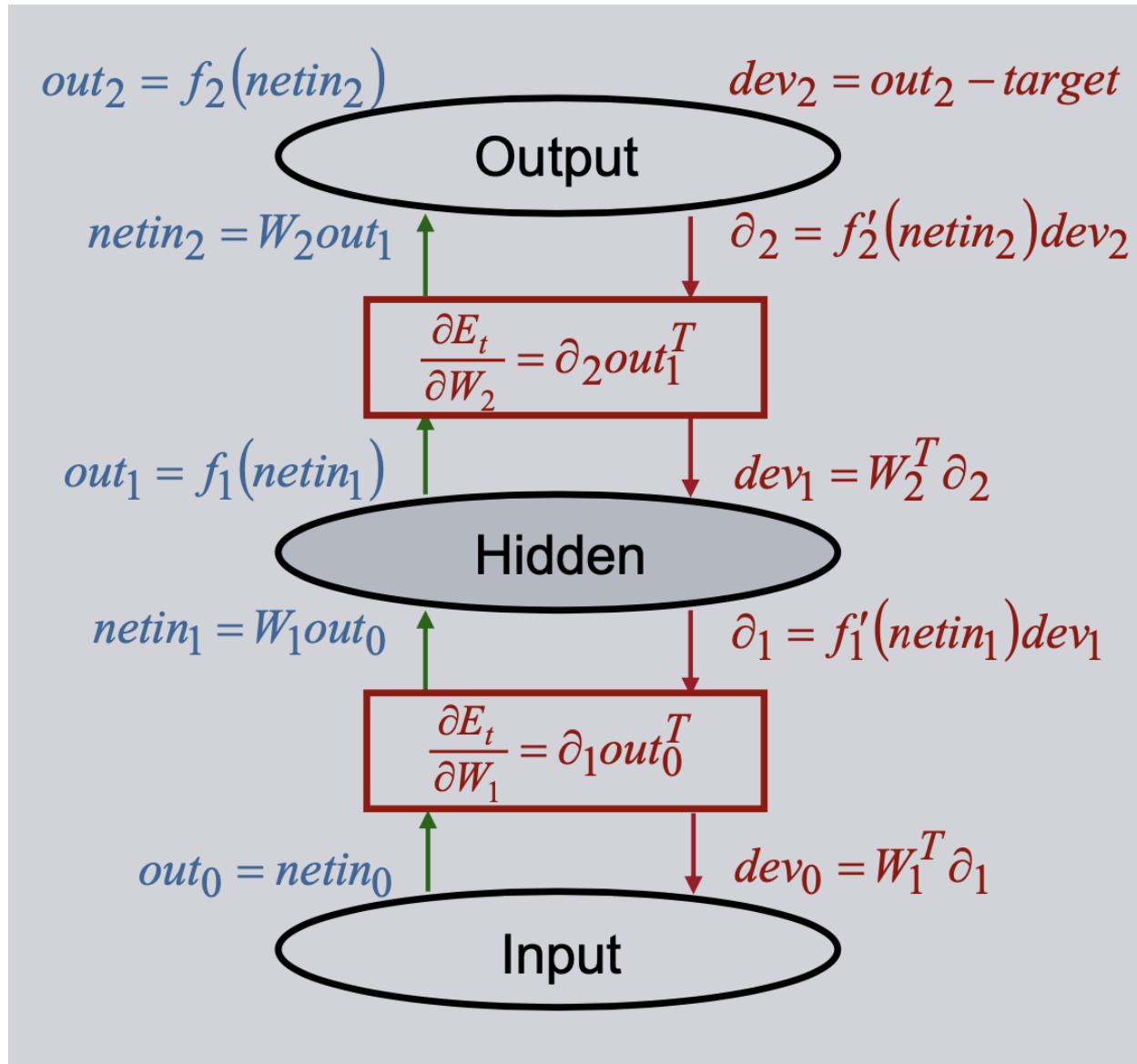
$\text{logistic}(x)$



$$\tanh'(0) = 1$$

$$(\text{logistic}'(0)) = \frac{1}{4}$$

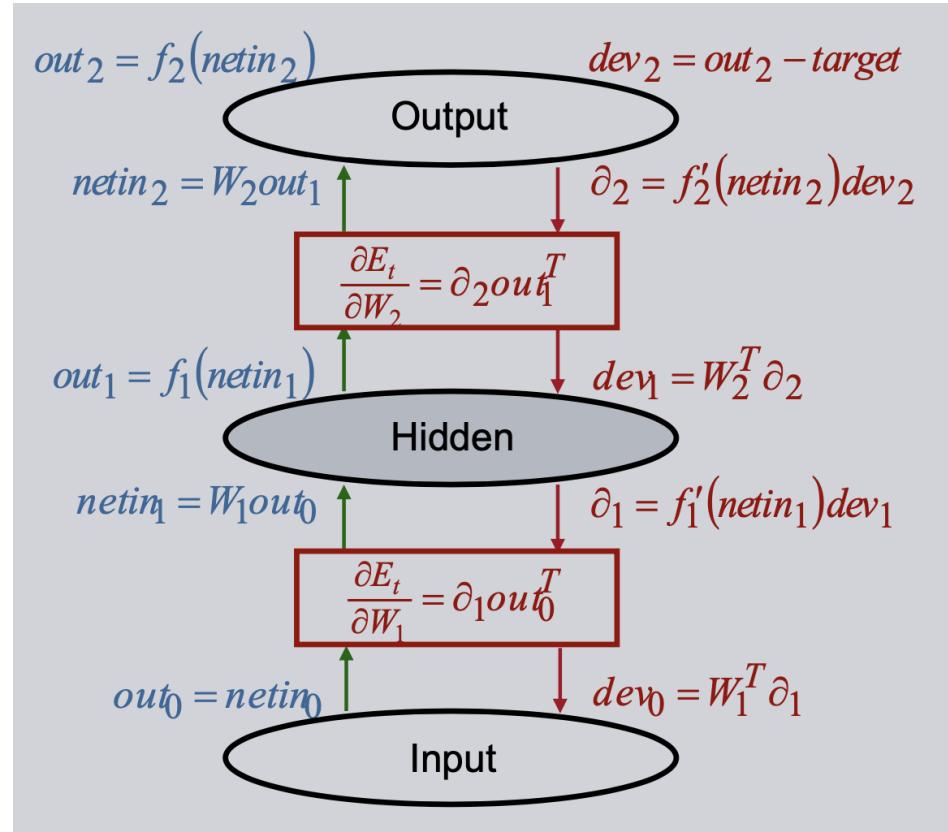
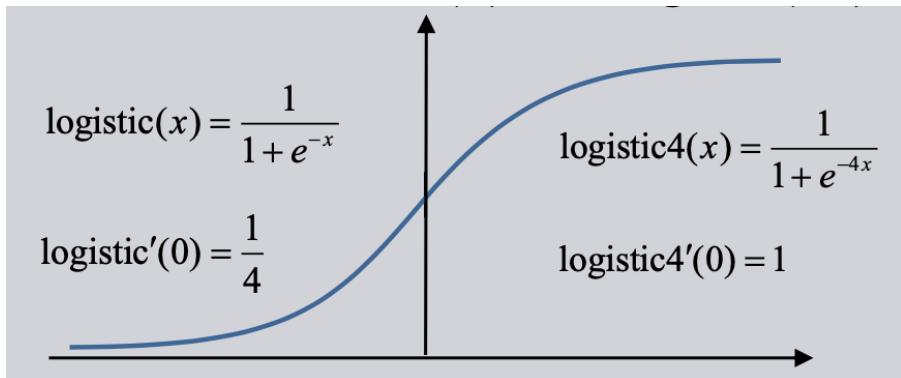
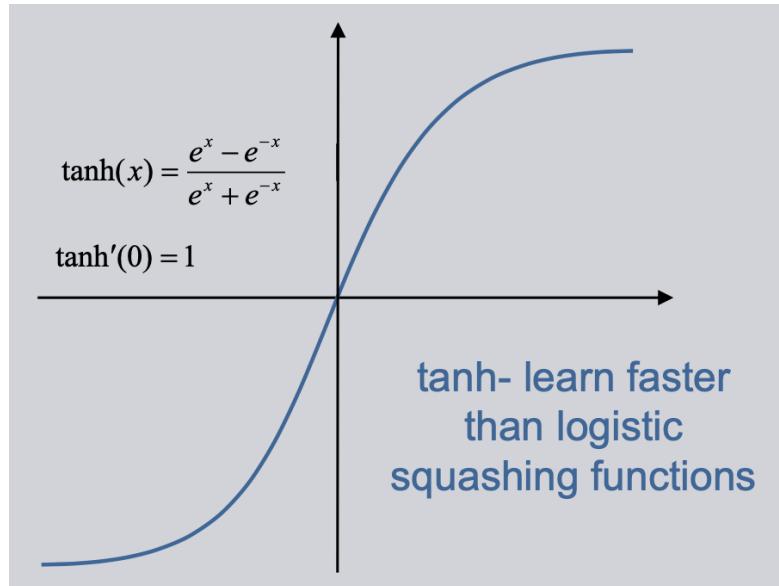
Beobachtung:



Die Ableitung der Aktivierungsfunktion beeinflusst den Fehlerfluss im Rückwärtspfad!

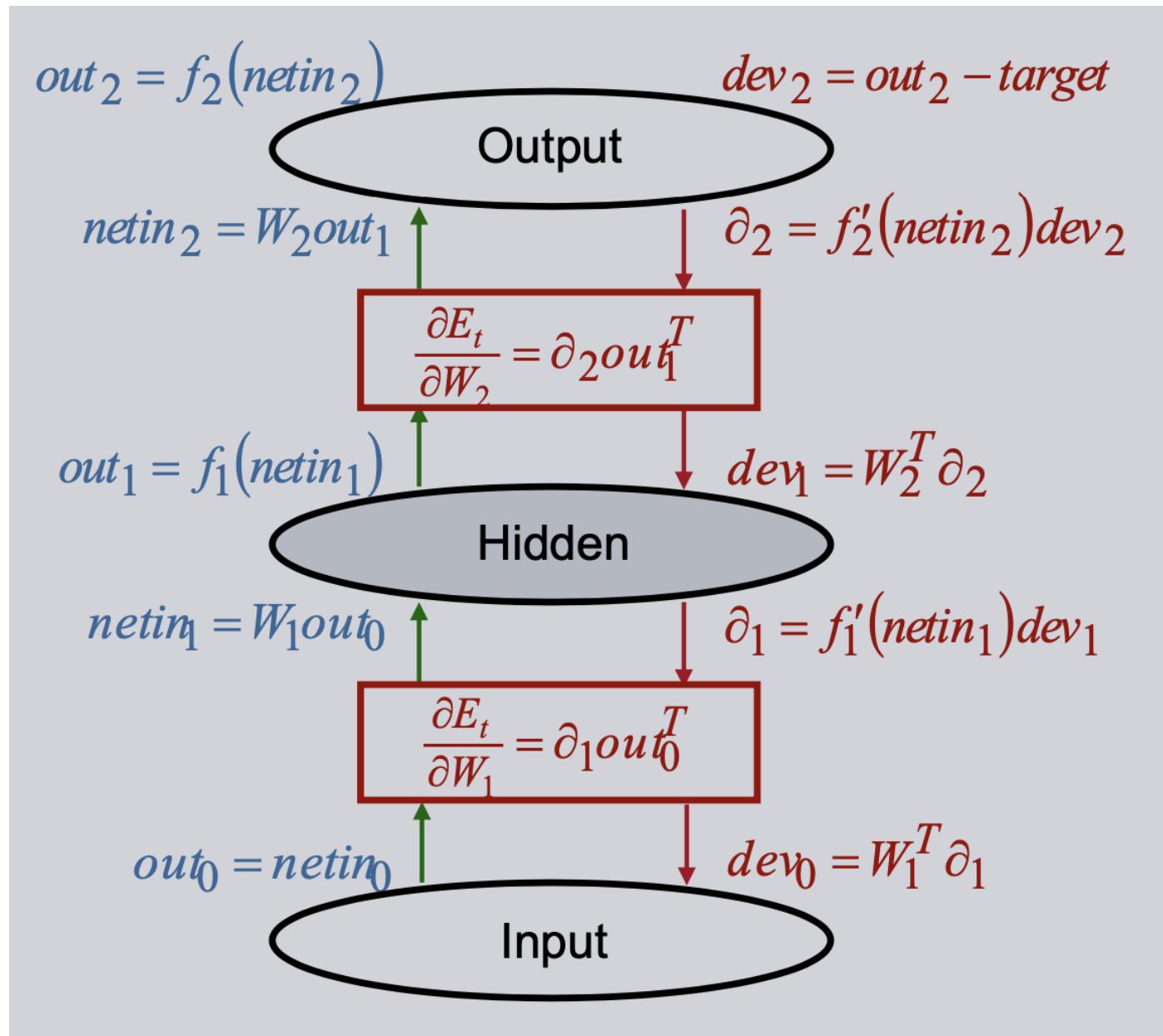
$\Rightarrow dev_2 \setminus dev_1$ werden mit f'_2 , f'_1 skaliert!

... für die Modellierung sind $\tanh(x)$ und $\text{logistic}(x)$ äquivalent.



... wegen $\tanh'(0) = 1 \neq \frac{1}{4} = \text{logistic}'(0)$

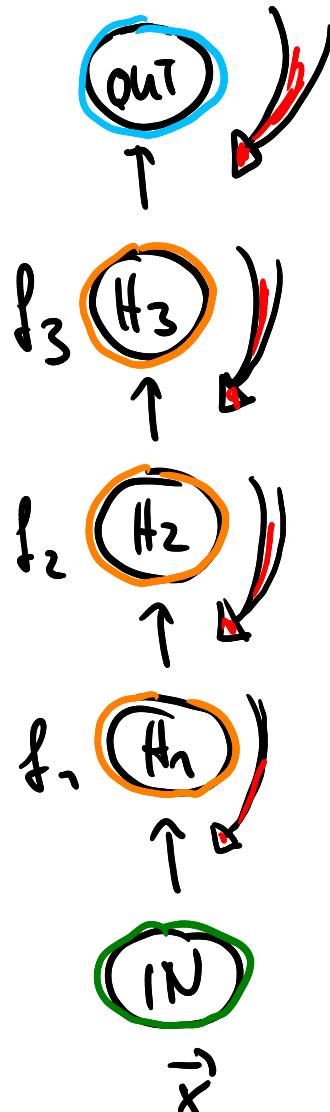
gilt das nicht fürs Lernen!!



- hier tanh:
 $f'_1(\text{nahe } 0) \approx 1$
- hier Logistic:
 $f'_1(\text{nahe } 0) \approx \frac{1}{4}$

⇒ Fehler wird gedämpft !
⇒ w_1 lernt langsamer

PROBLEM



- das "von unten" kommende Signal ist zwar betragsmäßig in der Regel klein, aber nicht exakt = 0
 $\Rightarrow \tanh'(\dots) < 1$

\Rightarrow der von oben kommende Fehler wird bei der Backpropagation bei jedem Durchgang durch eine Hidden-Schicht (leicht) gedämpft \rightarrow unten kommt nichts mehr an!



VANISHING - GRADIENT
PROBLEM

DEEP FFNN



gibt es Alternativen?

ReLU = Rectified Linear Functions

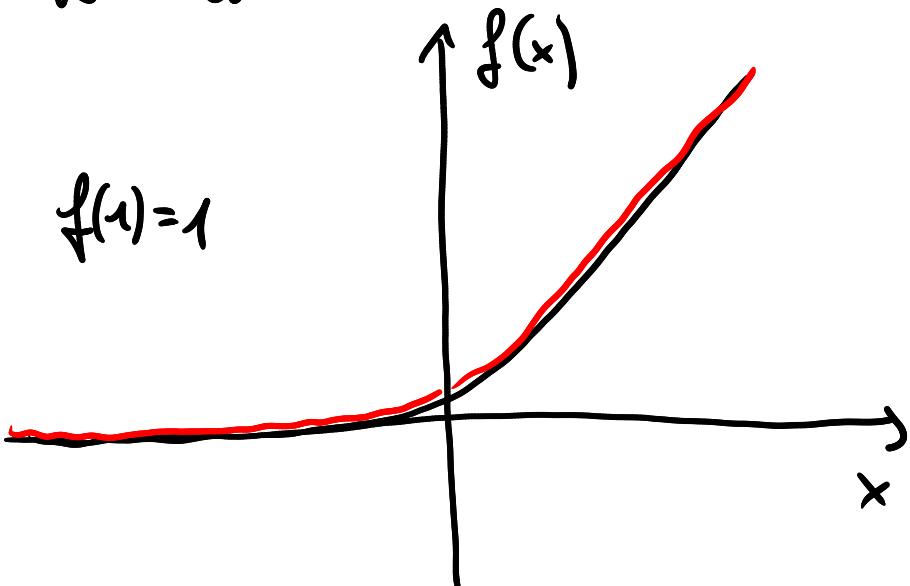


$$y = \begin{cases} x & \text{für } x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

- für $x > 0$: $f'(x) = 1$
⇒ keine Dämpfung des Fehlerflusses
- für $x < 0$: $f'(x) = 0$
⇒ überhaupt kein Fehlerfluss
- für $x = 0$: nicht differenzierbar
⇒ ? ?

Die Verwendung der ReLU -Funktion
behebt das Vanishing-Gradient Problem bei
tiefen Netzen!

Die Unstetigkeitstelle kann durch Glättung behoben
werden



zsp.:

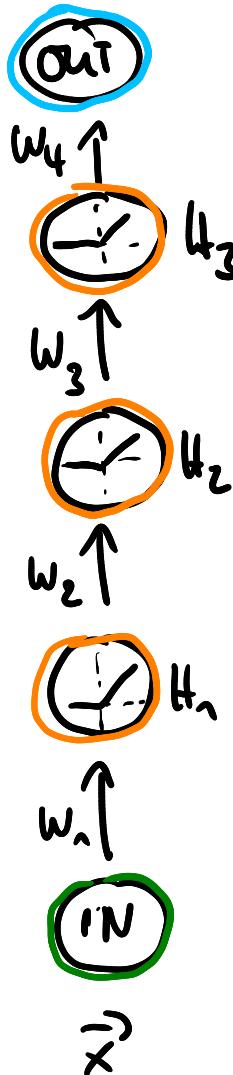
$$f(x) = \frac{1}{p} \ln(1 + e^{px})$$

$$f'(x) = \frac{1}{1 + e^{-px}}$$

$p \approx 0,63$ sorgt für
max. Ähnlichkeit

ÄBER:

VORSICHT beim Vorwärtspfad !!



- für $x_i \in (-1, 1)$ und $w_i \in (-r, r)$
 - es gibt negative Inputs für H_1
 - ReLU setzt diese auf 0
 - ⇒ Informationsverlust
- bei großen Gewichten in den Matrizen $W_1, W_2 \dots$ kommen (im realistisch) große Signale zu
 - ⇒ ReLU gibt diese unskaliert weiter!
 - ⇒ Informationsfluss wird nicht gedämpft...



Die Aktivierungsfunktion spielt für das Lernen eine zentrale Rolle!

- Unterschiedliche Aktivierungsfunktionen haben unterschiedliche Eigenschaften im Bezug auf den Informationsfluss im Netz
- das gilt sowohl im Bezug auf den Vorwärtspfad als auch auf den Rückwärtspfad

5.6) Occam vs. Bayes - bestes Modell vs. beste Lösung

(339)

→ START: Regressionsaufgabe (nicht-lineare Optimierung)
viele Features X_1, \dots, X_p

out
 $w_2 \uparrow$ 3-layer MLP

hid
 $w_1 \uparrow$ → trainiere ein Netz

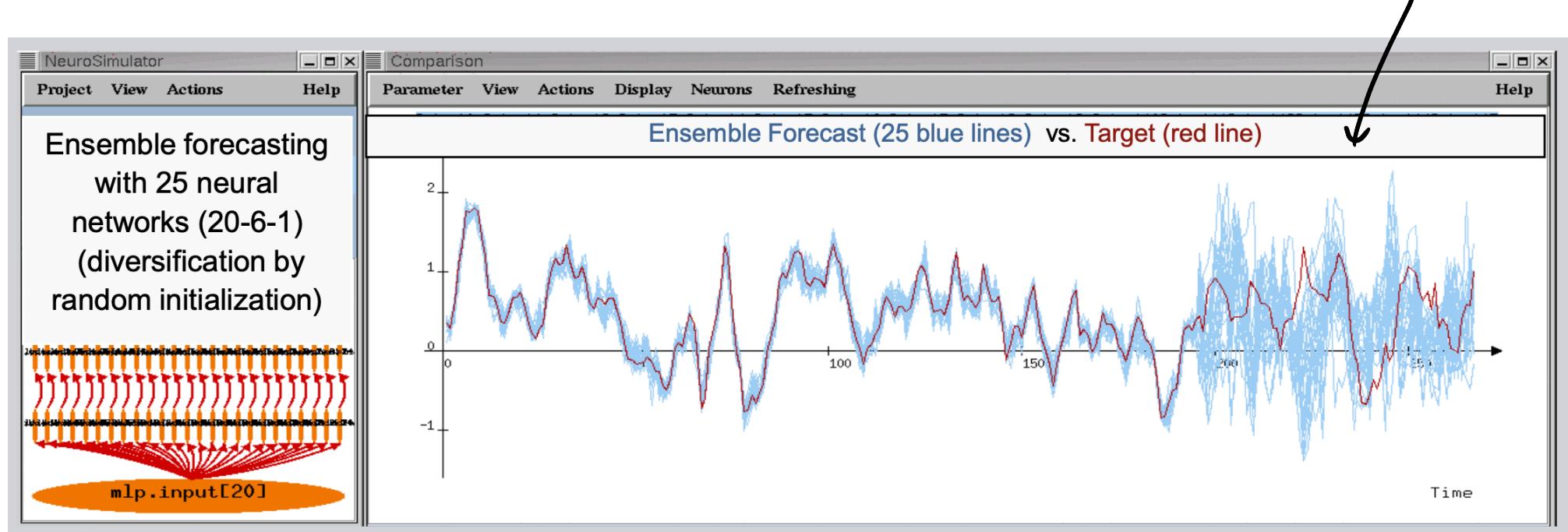
IN
• für eine optimale Lösung im Sinne des
"Universal Approximation Theorems" muss die
Hidden-Layer u.U. sehr groß sein.

DANN: sehr gutes Ergebnis auf der Trainingsmenge, Test: Fehler ok

ABER

Wiederholt man das mehrfach mit dem identischen Modell, so ergeben sich völlig unterschiedliche Prognosen auf der Testmenge!

25 Modelle

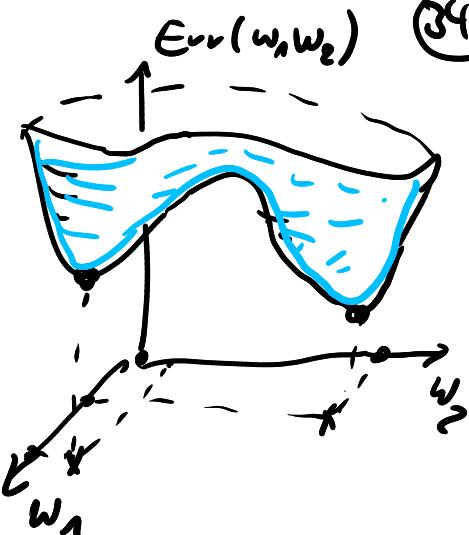


FRAGE

Was passiert hier?

Erklärungsversuche:

(341)



- ① unterschiedliche Initialisierung der Gewichtsmatrizen führt beim Lernen dazu, dass wir im verschiedenen lokalen Minima der Fehlerfläche landen

⇒ komplett unterschiedliche Belegung von

w_1, w_2

⇒ verschiedene Predictions mit (ähnlichem) Fehler

- ② zu viele Parameter im Verhältnis zur Anzahl der Trainings-Samples (Faustregel: 10x so viele Samples wie Parameter)

... was stimmt hier denn jetzt? ① oder ②?

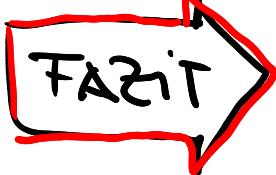
Überlegung:

→ würde ① gelten, dann würden sich auch die Predictions auf der Trainingsmenge deutlich unterscheiden!

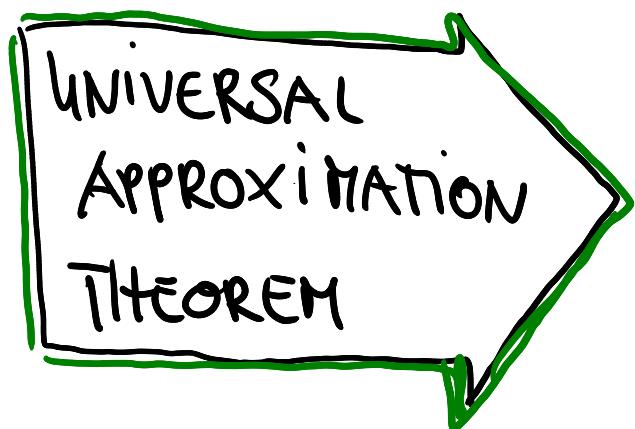
Das ist hier aber nicht der Fall.

→ Erklärungsversuch ② trifft wohl eher zu!

Bemerkung: Das Problem der verschiedenen Lösungen durch Finden verschiedener lokaler Minima trifft tatsächlich eher bei Modellen mit wenigen Variablen zu. (allg. Beobachtung)



- "kleine" Modelle (wenige Features, wenig Modellparameter) leiden unter dem Problem der lokalen Minima (Initialisierung)
- "große" nicht-lineare Modelle haben aber ein Problem wegen ihrer starken Überparametrisierung.

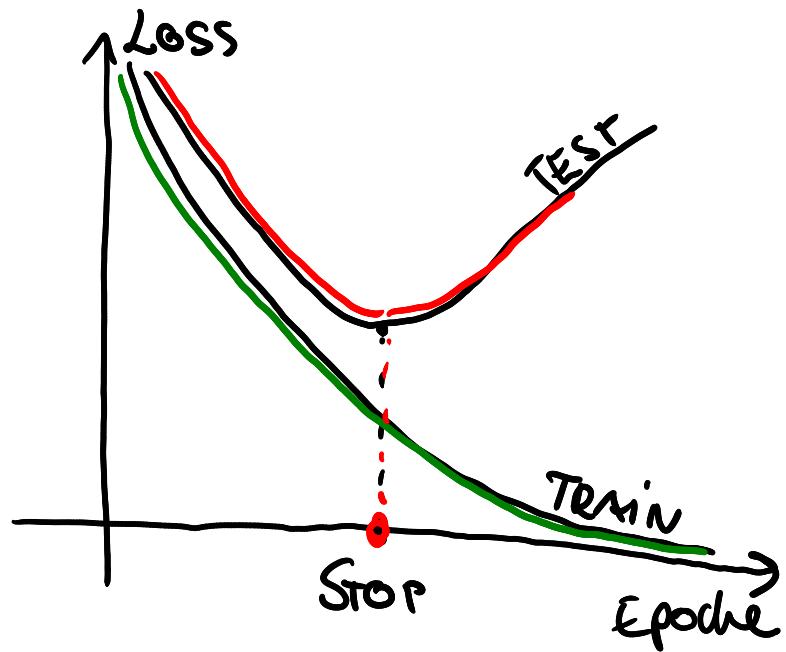


5.6.1) Regularisierungverfahren bei Neuronalen Netzen

344

... einfachste Lösung für ein festgelegtes Setting der Modellierung

① EARLY STOPPING



/ das haben wir so ähnlich
schon bei Decision Trees etc.
gesehen

← splitte die verfügbaren Daten in
Trainings- und Testmenge

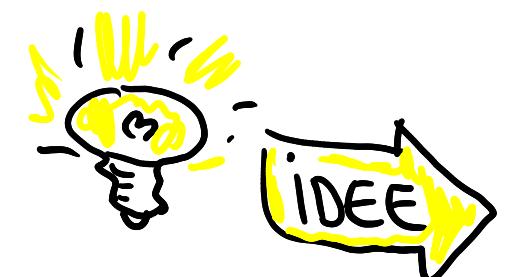
- beobachte den Trainings- und den Testfehler während des Lernens

- verwende das Modell mit dem geringsten Testfehler bzw. stoppe sobald dieser wieder ansteigt.

⇒ Modellkomplexität ist fix!
keine "echte" Regularisierung ...

② Square Weight Decay

ERINNERUNG: LASSO \approx sorgt dafür, dass einige Parameter des Modells auf Null gesetzt werden (verschwinden)



Erzwinge kleine Gewichte in einigen der Korrektoren (idealerweise = 0 !)

Analogie LASSO: füge einen Penalty-Term in der Fehlerfunktion ein, der kleine Gewichte erzwingt

$$E(w) = E_0(w) + \lambda \sum \frac{1}{2} w_i^2 \rightarrow \min_w$$

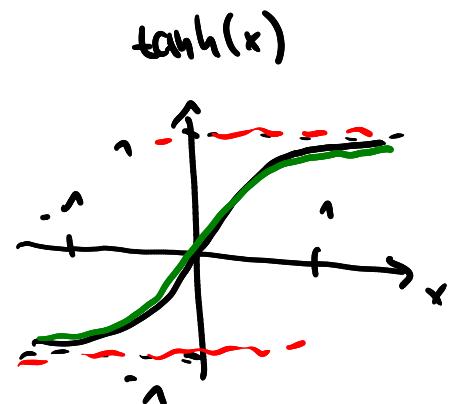
$$\text{When.}_i = w_{\text{alt},i} - \eta \frac{\partial E}{\partial w_i} = w_{\text{alt},i} - \eta \left(\frac{\partial E_0}{\partial w_i} + 2\lambda w_i \right)$$

$\lambda \neq 0$
zielt das Gewicht in Richtung 0 !!

der weight decay sorgt also dafür, daß die Gewichte stetig verkleinert werden, es sei denn die "neue" Fehlerfunktion steuert dagegen.



- wie soll man λ wählen?
- für kleine Gewichte landet man im linearen Bereich der $\tanh(\dots)$ -Aktivierung ↗ ↗



\Rightarrow Weight Decay hat einen ausgeprägten Prior hin zu linearen Modellen!

... aber das wollten wir ja gerade nicht haben 😞

5.6.2) Occam's Razor

347

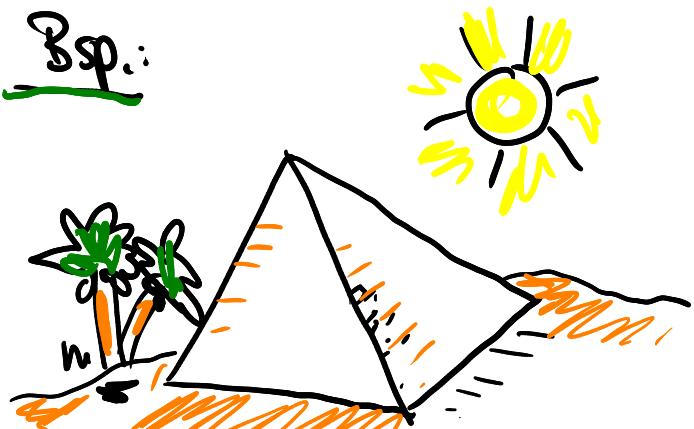
nach Wilhelm von
Ockham (1288 - 1347)

heuristisches Forschungsprinzip :

Bei der Bildung von erklärenden Hypothesen
und Theorien ist Sparsamkeit geboten !



Bsp.:



Wer hat die Pyramiden gebaut ??
Und wie ??

- Ägypter
- Aliens
- Atlanten

- ...
- ...

Occam's Razor:

- ① die einfachste von allen möglichen Theorien ist zu präferieren
- ② eine Theorie ist genau dann einfach, wenn sie möglichst wenige Unbekannte \ Variablen und Hypothesen enthält
- ③ die Variablen und Hypothesen sollten im klaren logischen Beziehungen zueinander stehen

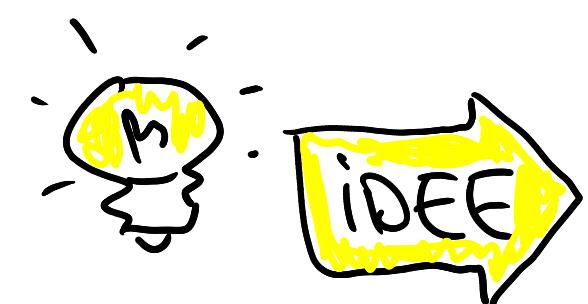


→ unsere Modelle, die neuronalen Netze, sind drastisch überparametrisiert !:



Wie werde ich möglichst viele Modellparameter los?

→ "weight decay" wäre eine Möglichkeit, aber hier führen wir ja einen prior zum zu linearen Modellen ein (→ andere Modellklasse!)



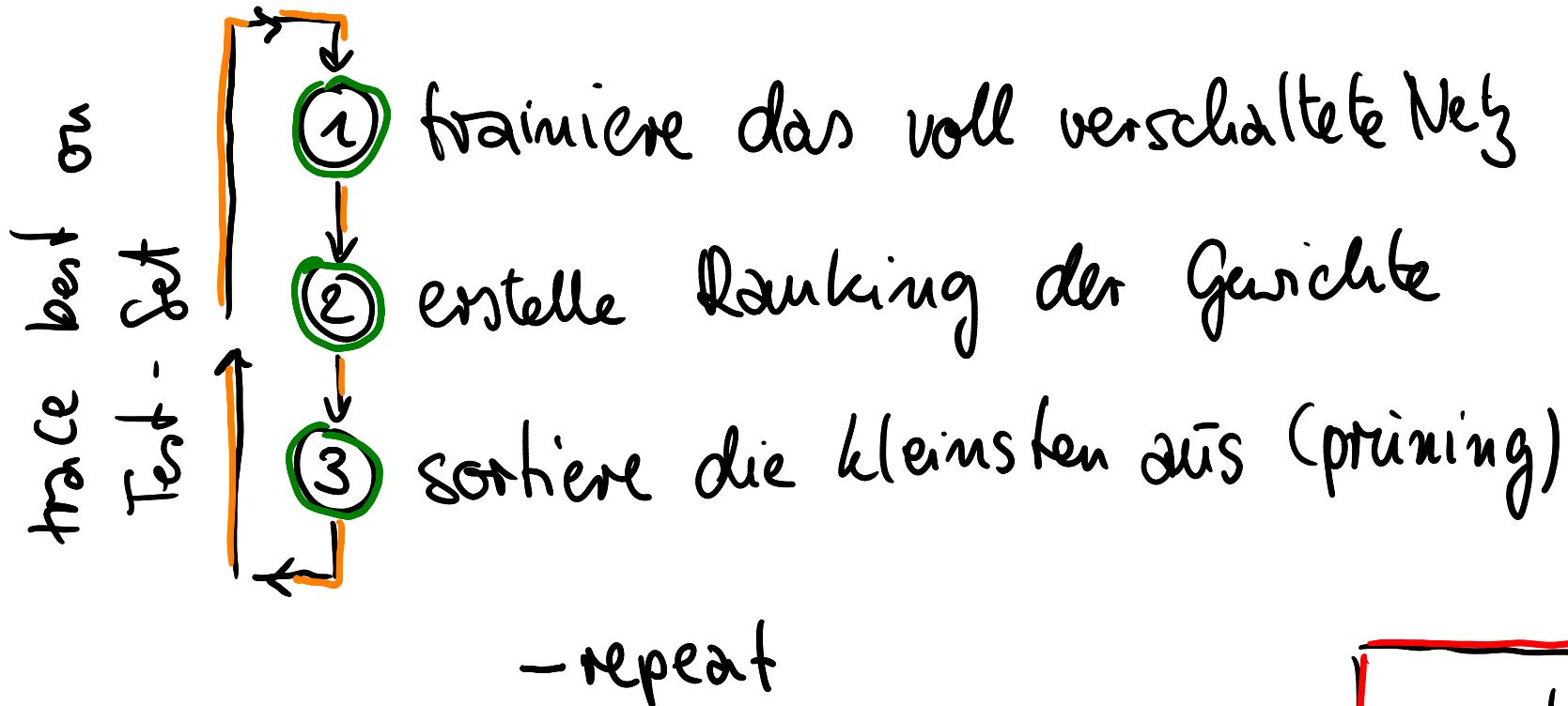
Kasse das überparametrisierte Modell
einmal lernen und sorte dann die
unwichtigen Gewichte aus.

② Kriterium für die "Wichtigkeit" eines Gewichtes?

- Größe des Gewichtes, also $|w_{ij}|$ oder
- w_{ij}^2
- ...

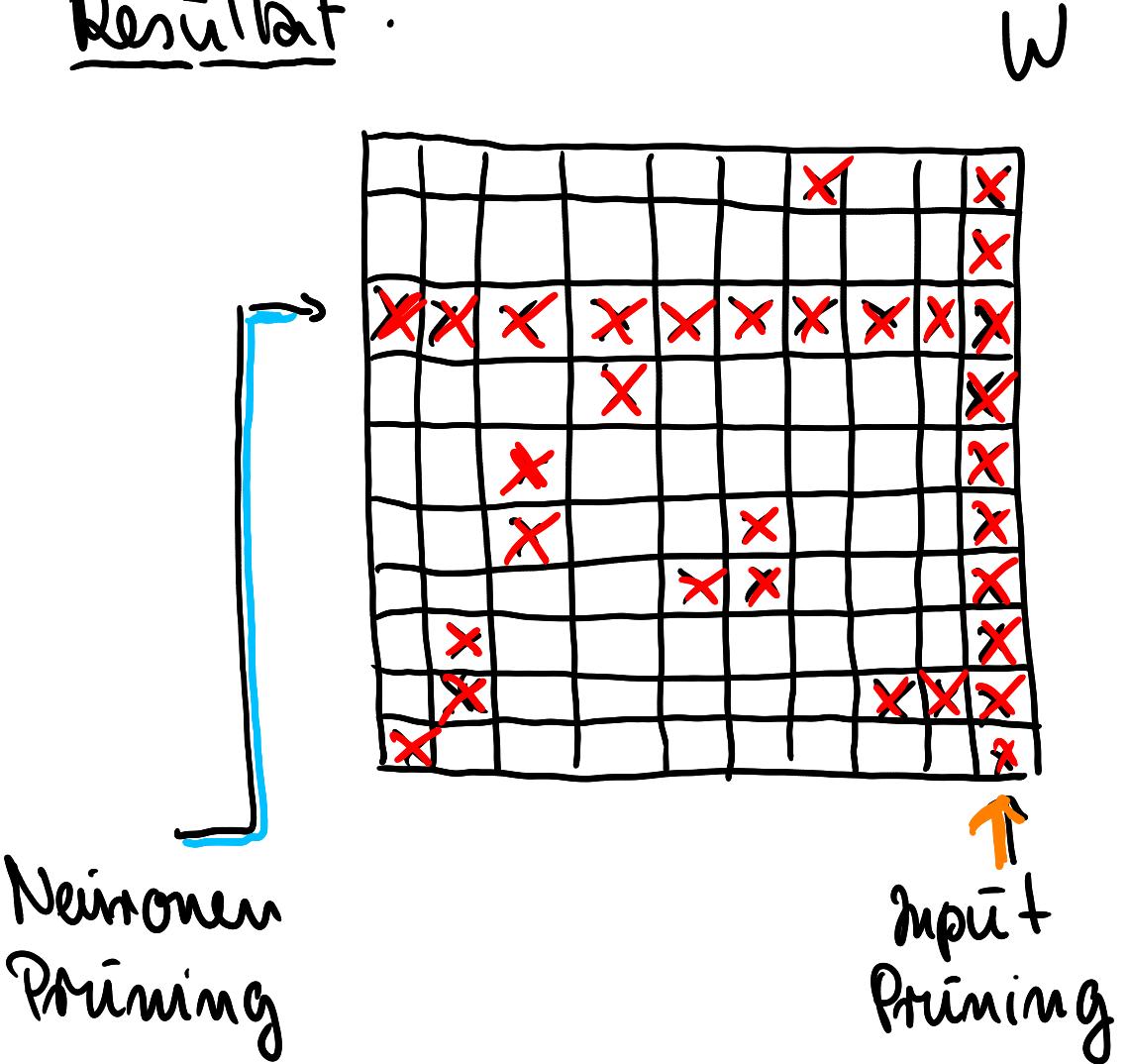
⇒ Weight Pruning

Weight Pruning Algorithmus



mathematische
Umsetzung von
Occam's Razor

Resultat:



Neintonen Priming

- Kriterium für Wichtigkeit?
 - wieviele aussortieren pro Runde

Sparse Matrix

"dünne" Verschaltung

VORTEIL

kein prior auf
lineare Modelle

NACHTEIL

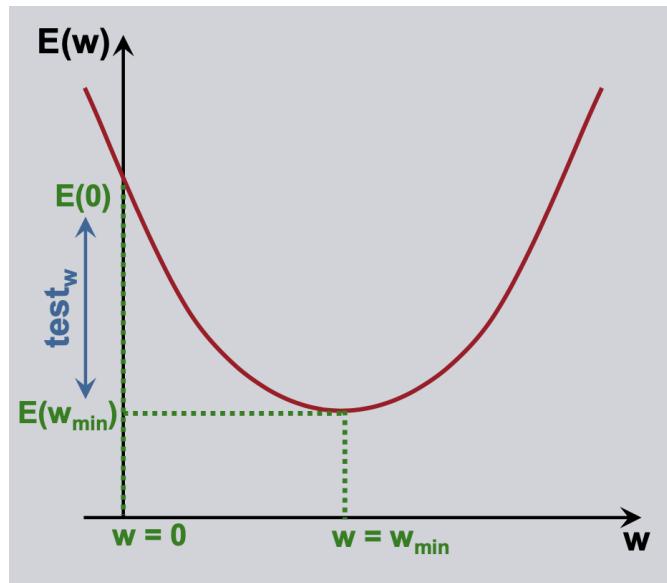
Bastelstunde

- Train\Test Split ?

Alternativen?

Ein Beispiel:

: "Optimal Brain Damage" (OBD) (LeCun et al. 1990)



Idee Augenommen, wir hätten ein Minimum der Fehlerfunktion $E(w)$ gefunden: welche Gerichte kann man auf Null setzen, ohne den Fehler zu stark zu stören?

Taylorentwicklung $E(w)$ in einem lokalen Minimum: $w_{\min} = w$

$$\frac{\partial E(w)}{\partial w_{\min}} \approx 0$$

$$\Rightarrow E(0) \approx E(w) + \frac{\partial E}{\partial w}(0-w) + \frac{1}{2} \frac{\partial^2 E}{\partial w^2}(0-w)^2$$

||

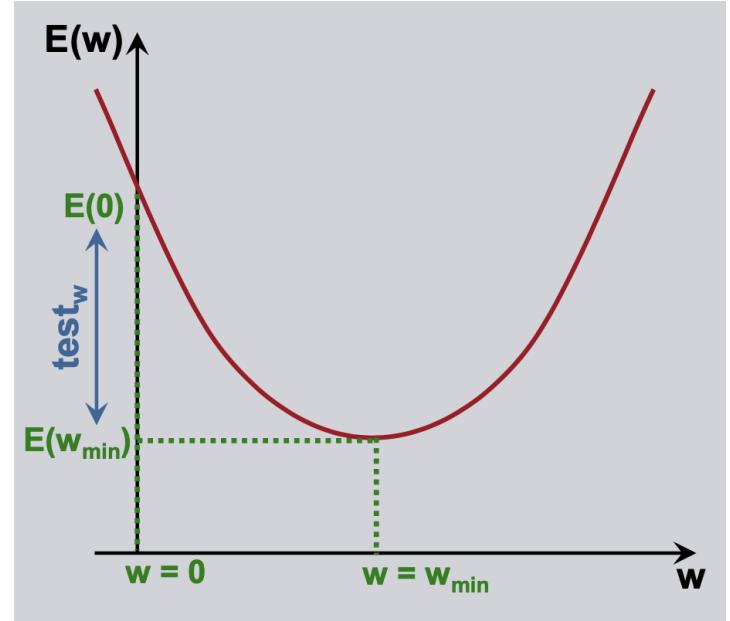
≈ 0 wg.
lok. Min

wie stark wächst $E(w)$, wenn man w in die Null verschiebt?

$$\text{test}_w = E(0) - E(w) \approx \frac{1}{2} \frac{\partial^2 E}{\partial w^2} \cdot w^2$$

\uparrow
Änderung bei Verschiebung

... noch viele andere Verfahren ...

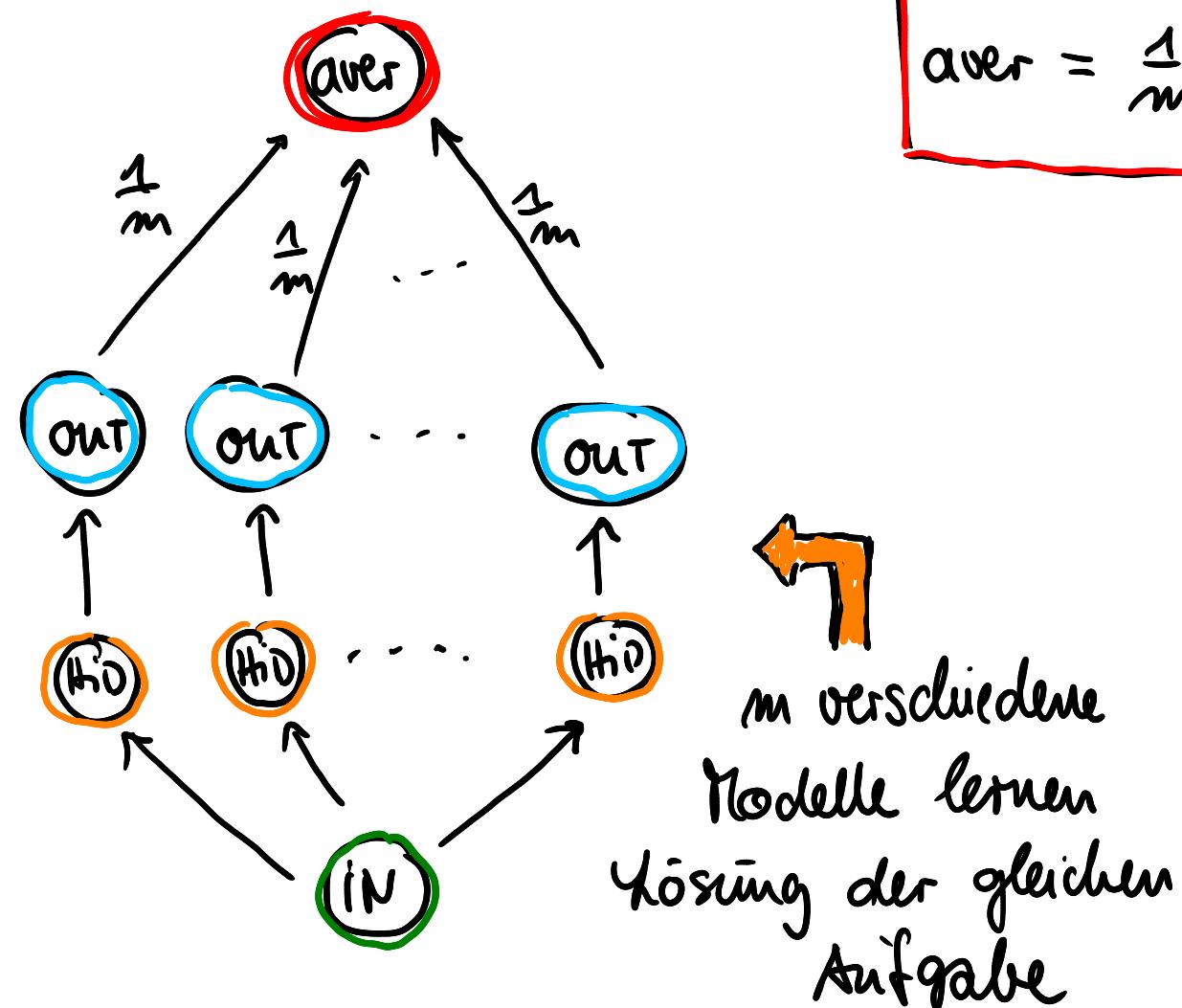


Scheide diejenigen w_{ij} mit den kleinsten $\text{test}_{w_{ij}}$

und noch eine letzte Bemerkung zu FFNNs

5.6.3) Ensembles

Reduktion des Fehlers durch Mittelwert!



$$\text{aver} = \frac{1}{m} \sum_{i=1}^m \text{OUT}_i$$

$$E_{\text{aver}} \approx \frac{1}{m} \text{aver}(E_i)$$

↑ falls die Fehler der einzelnen Modelle unkorreliert sind ...

Bias-Variance Dilemma:

$$y = f(x) + \varepsilon \quad \text{wahrer Modell}$$

$$y = f_i(x) \quad (i=1,\dots,m) \quad \text{einzelnes Modell}$$

sowie $y = \bar{f}(x) = \frac{1}{m} \sum_{i=1}^m f_i(x)$ ↗
Average

$$\begin{aligned} \frac{1}{m} \sum_{i=1}^m \frac{1}{T} \sum_t (y_t - f_i(x_t))^2 &= \dots = \\ &= \frac{1}{T} \sum_t (y_t - f_i(x_t))^2 + \quad \leftarrow \text{NOISE} \\ &+ \frac{1}{T} \sum_t (f_i(x_t) - \bar{f}(x_t))^2 + \quad \leftarrow \text{BIAS} \\ &+ \frac{1}{m} \sum_i \frac{1}{T} \sum_t (\bar{f}(x_t) - f_i(x_t))^2 \quad \leftarrow \text{VARIANZ} \end{aligned}$$

für $f_i \approx \bar{f}$: Bias klein + } \Rightarrow Gesamtfehler klein!
 Varianz klein }



PERFECT!!