Foto: Stripe Media auf Unsplash

Training von neuronalen Netzen

Üben, üben, üben

INNOVET
KI B³
Künstliche Intelligenz in die berufliche Bildung bringen

INNOVET
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
bibb Bundesinstitut für Berufsbildung

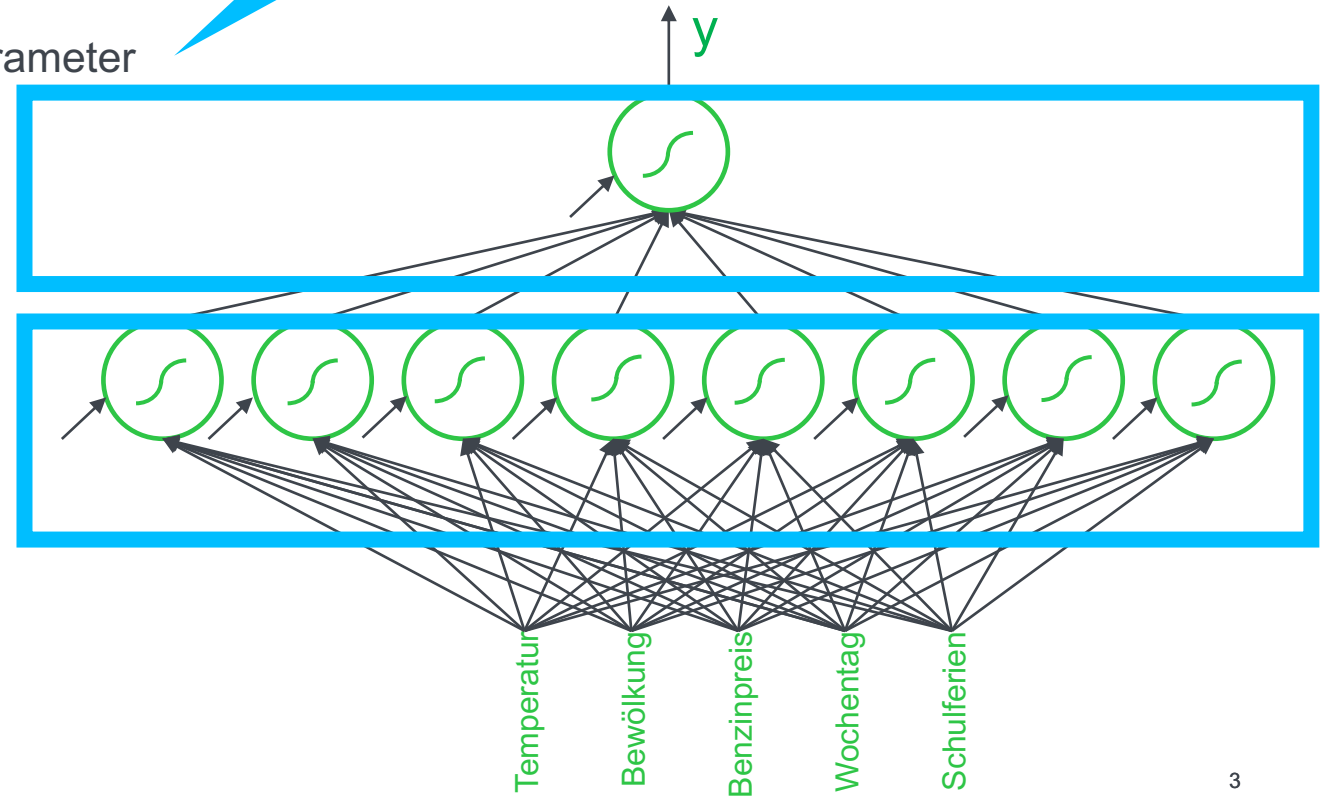
Gefördert als InnoVET-Projekt aus Mitteln des Bundesministeriums für Bildung und Forschung.

Ein Beispieltraining

Das Beispiel vom letzten Mal

Zu trainieren!

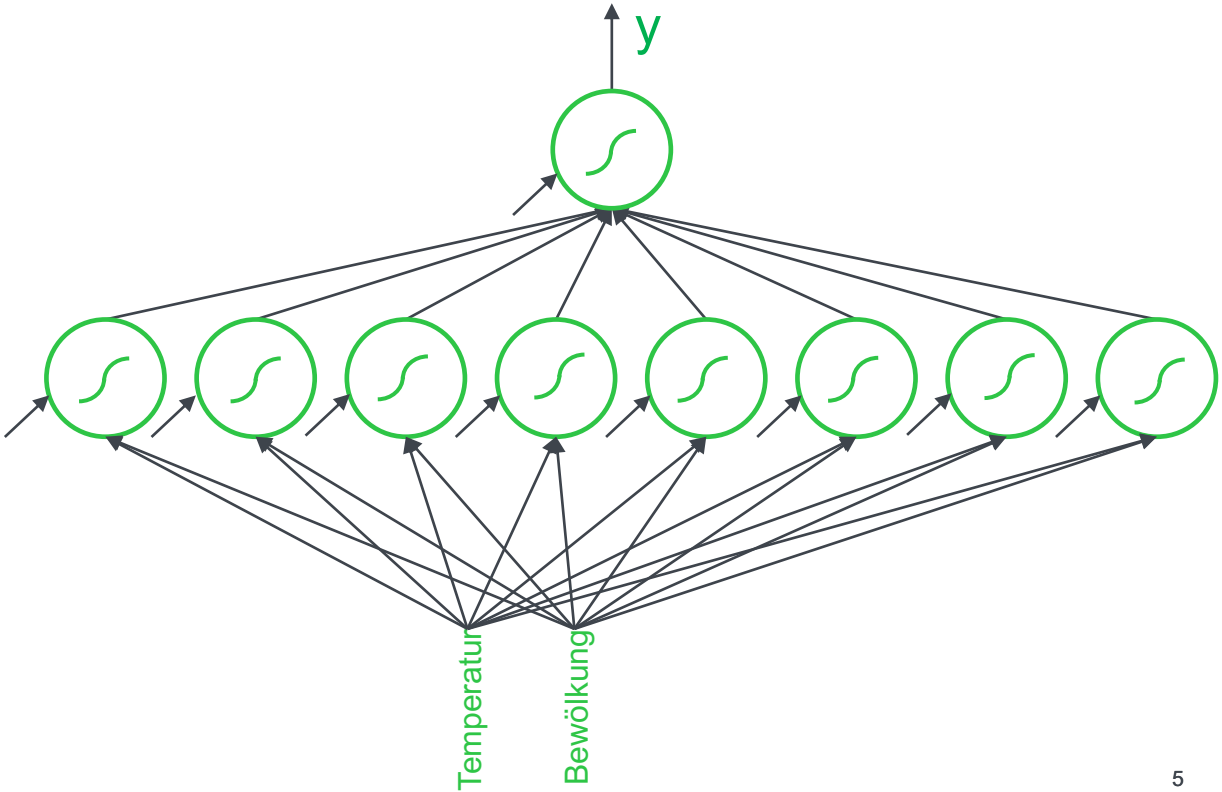
- Insgesamt 48+9=57 Parameter
- Schicht 2
  - 1 Neuron,
    8 Inputs, Bias
  - 1*9 = 9 Parameter
- Schicht 1
  - 8 Neuronen,
    5 Inputs, Bias
  - 8*6 = 48 Parameter



Drag and Drop: Welche Rechnung passt zu welchem Schritt?

Das Beispiel vom letzten Mal

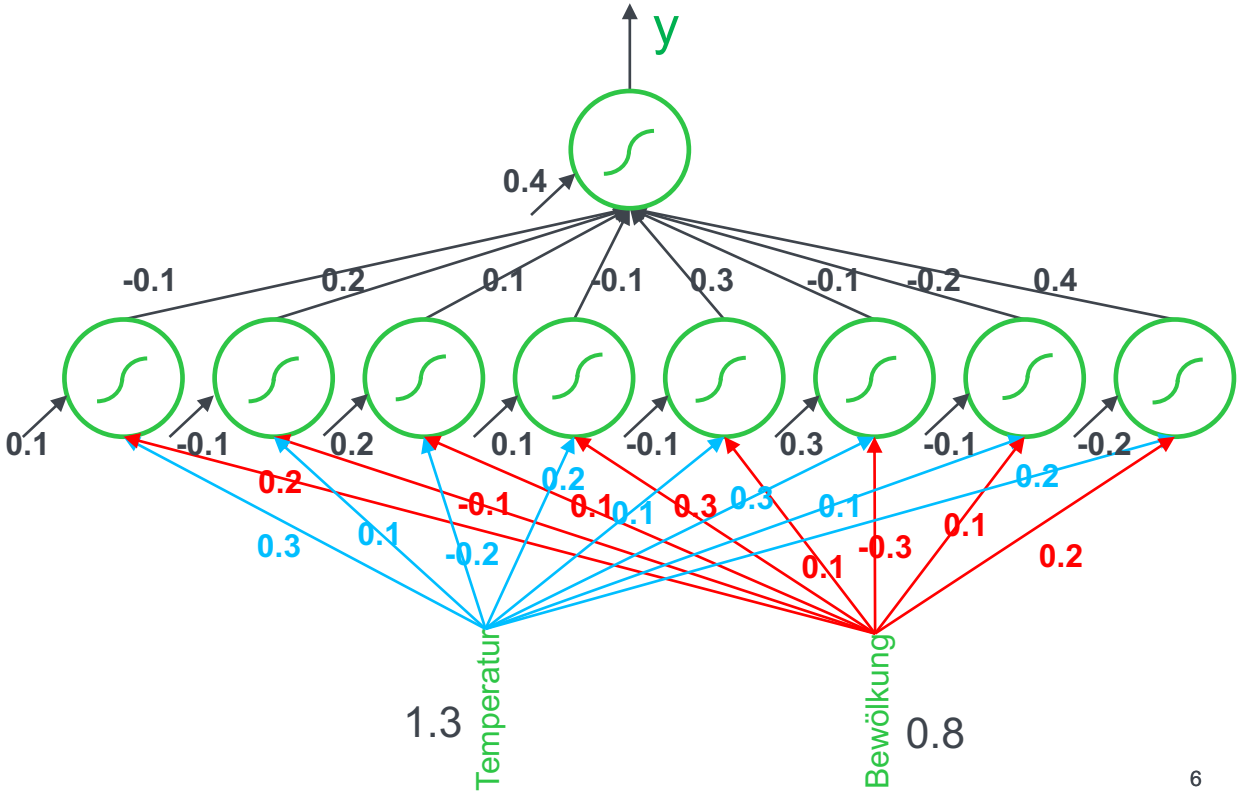
- Vereinfachung:
  nur 2 Inputs



Das Beispiel vom letzten Mal

- Beispieltraining
- Initialisierung mit zufällig gewählten Parametern
- Ausprobieren, wie gut das Netz ist:
  Forward Pass für einen Datenpunkt
  - Temperatur 1.3 (x_1)
  - Bewölkung 0.8 (x_2)

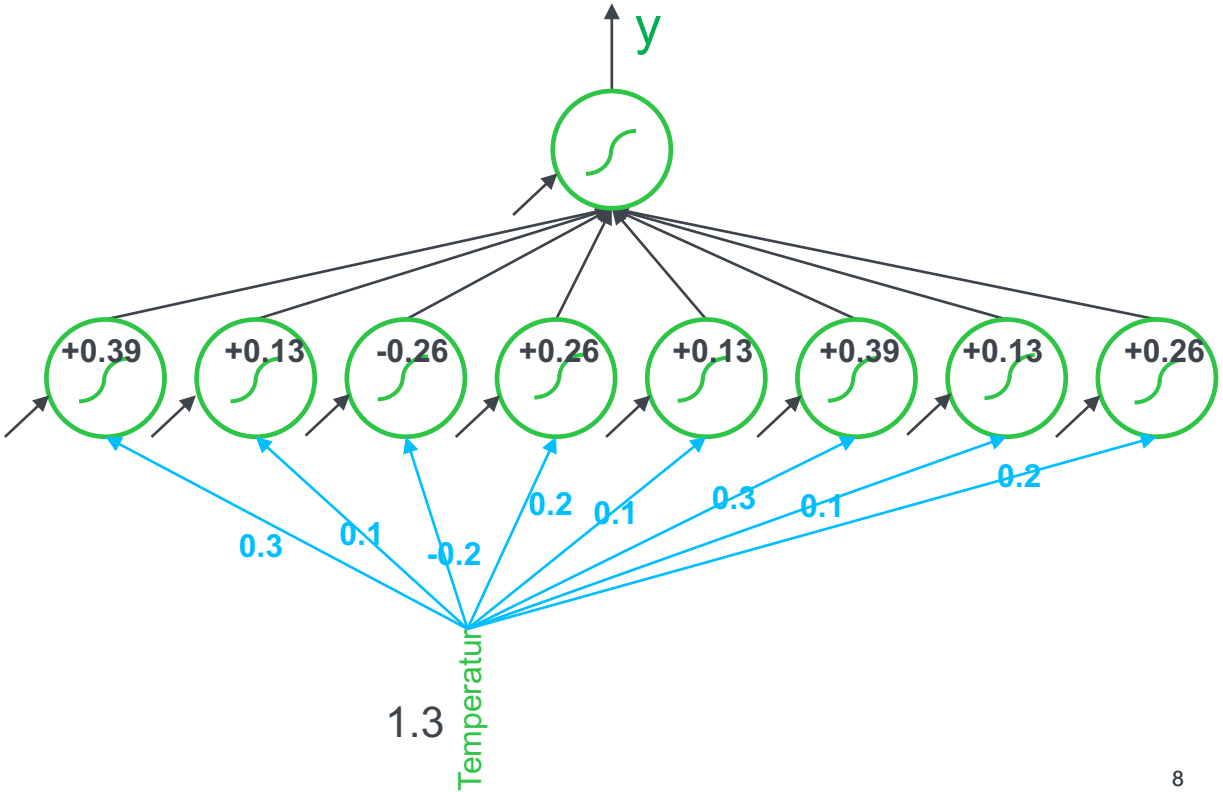
  (standardisierte Werte)



Multiple Choice: Abweichung

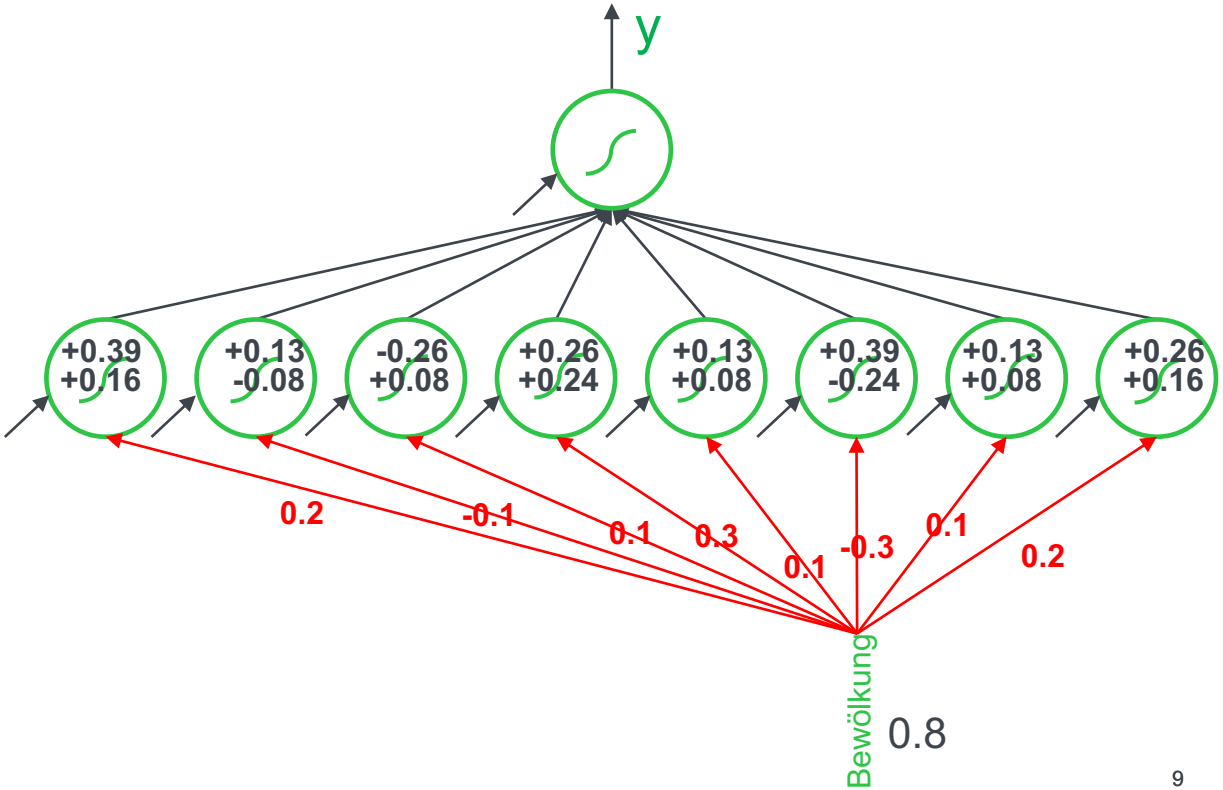
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur



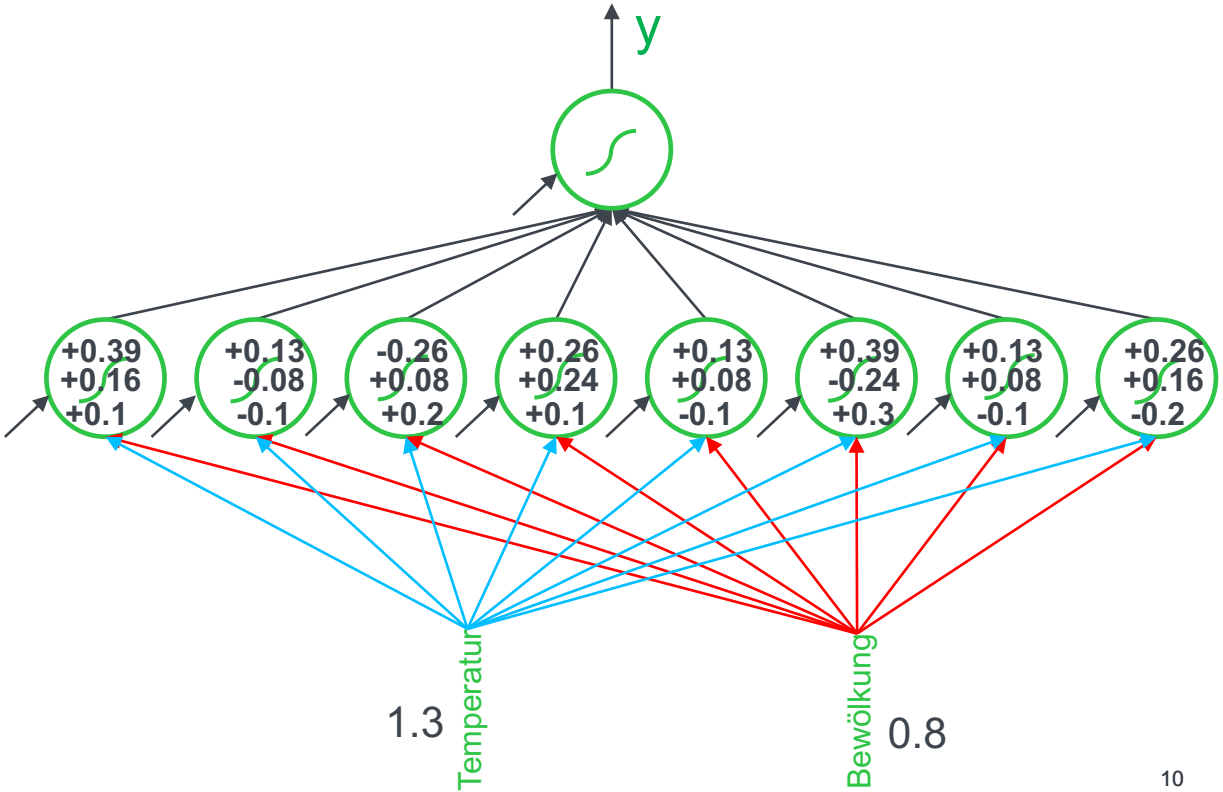
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung



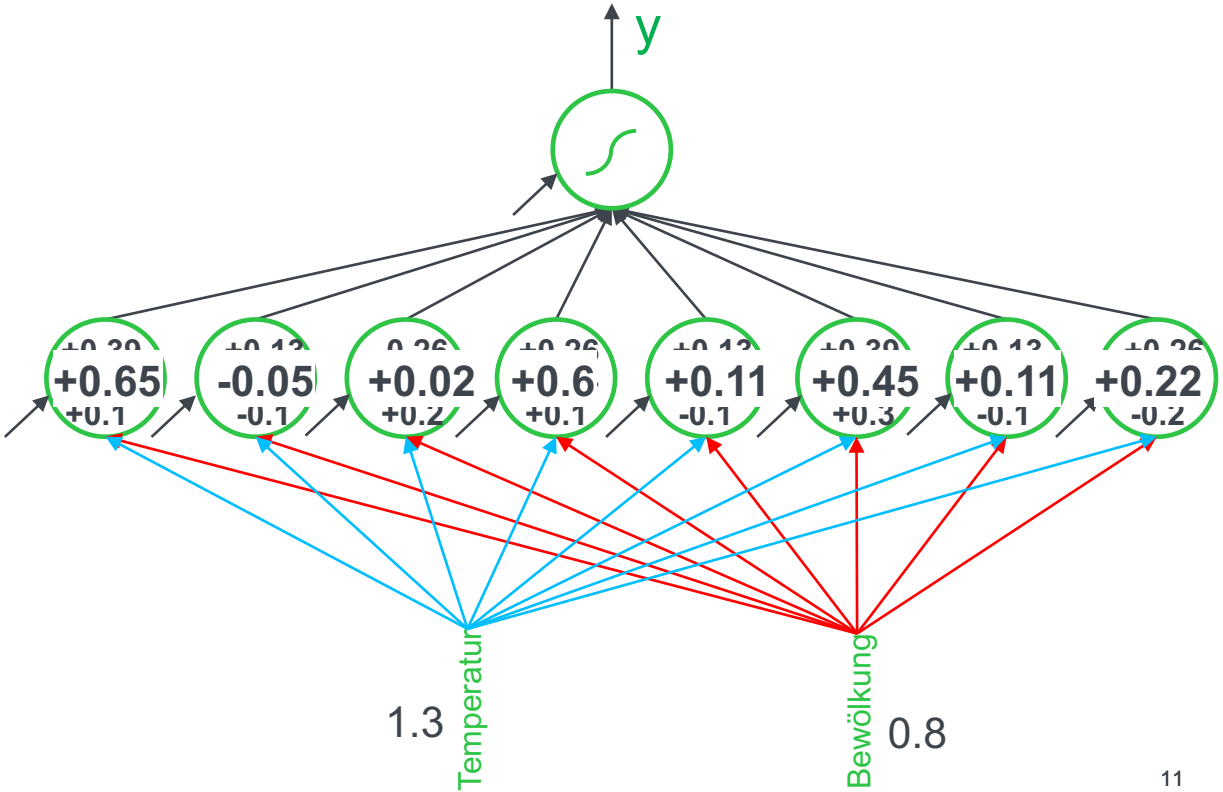
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung
  - Bias



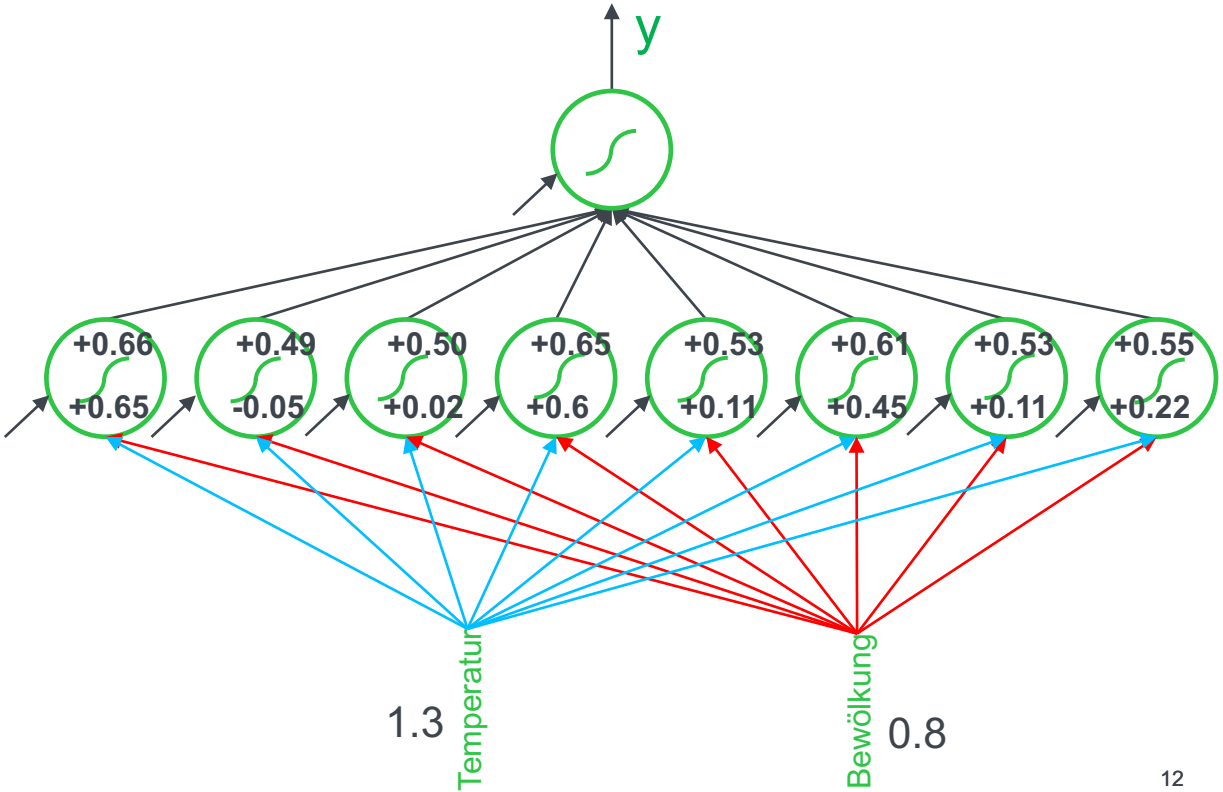
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung
  - Bias



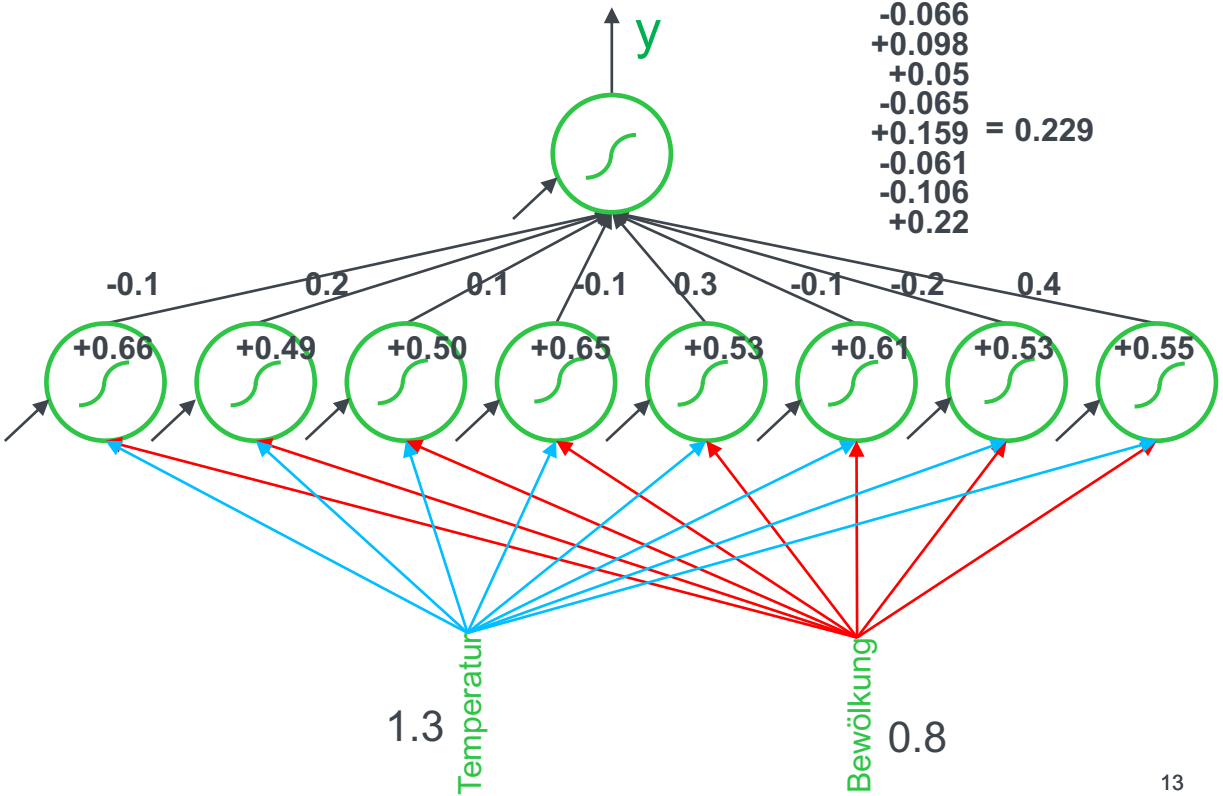
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung
  - Bias
- Sigmoid-Aktivierung der ersten Schicht (a_i^1)



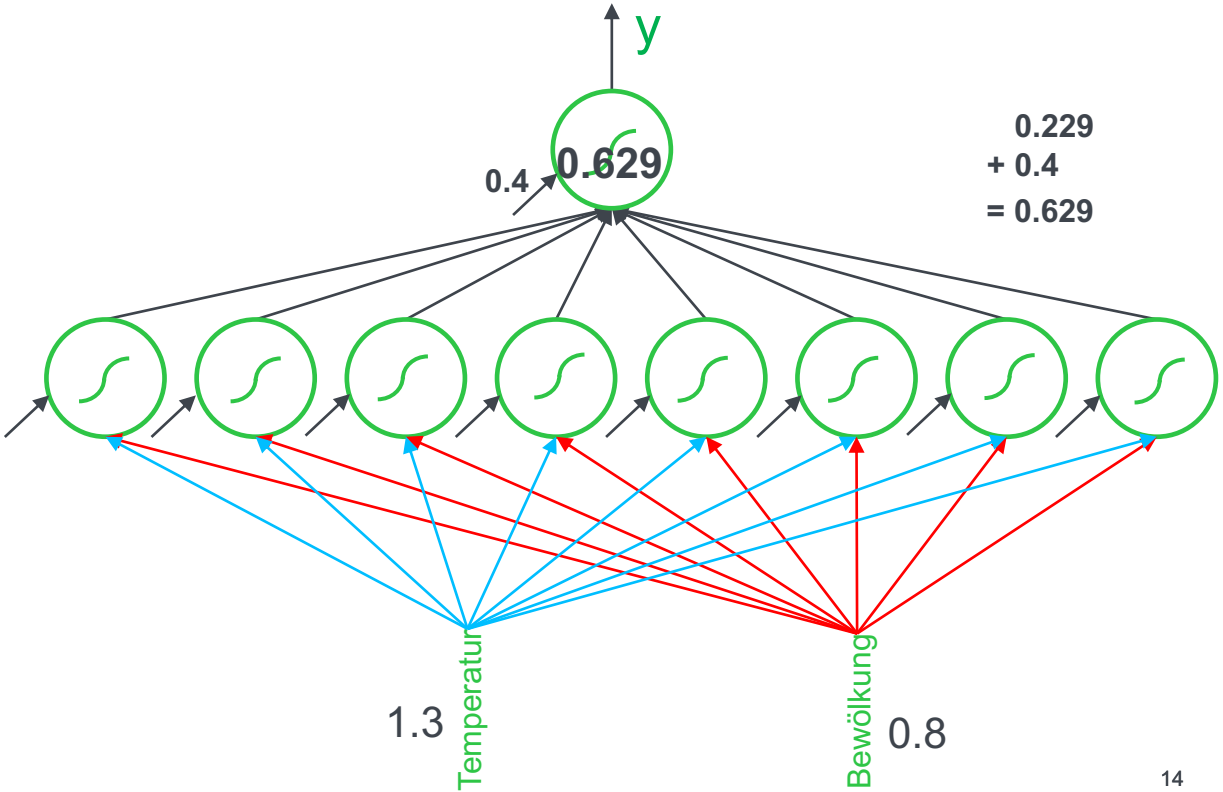
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung
  - Bias
- Sigmoid-Aktivierung der ersten Schicht (a_i^1)
- Logits der zweiten Schicht (z_i^2)
  - Inputs von Schicht 1



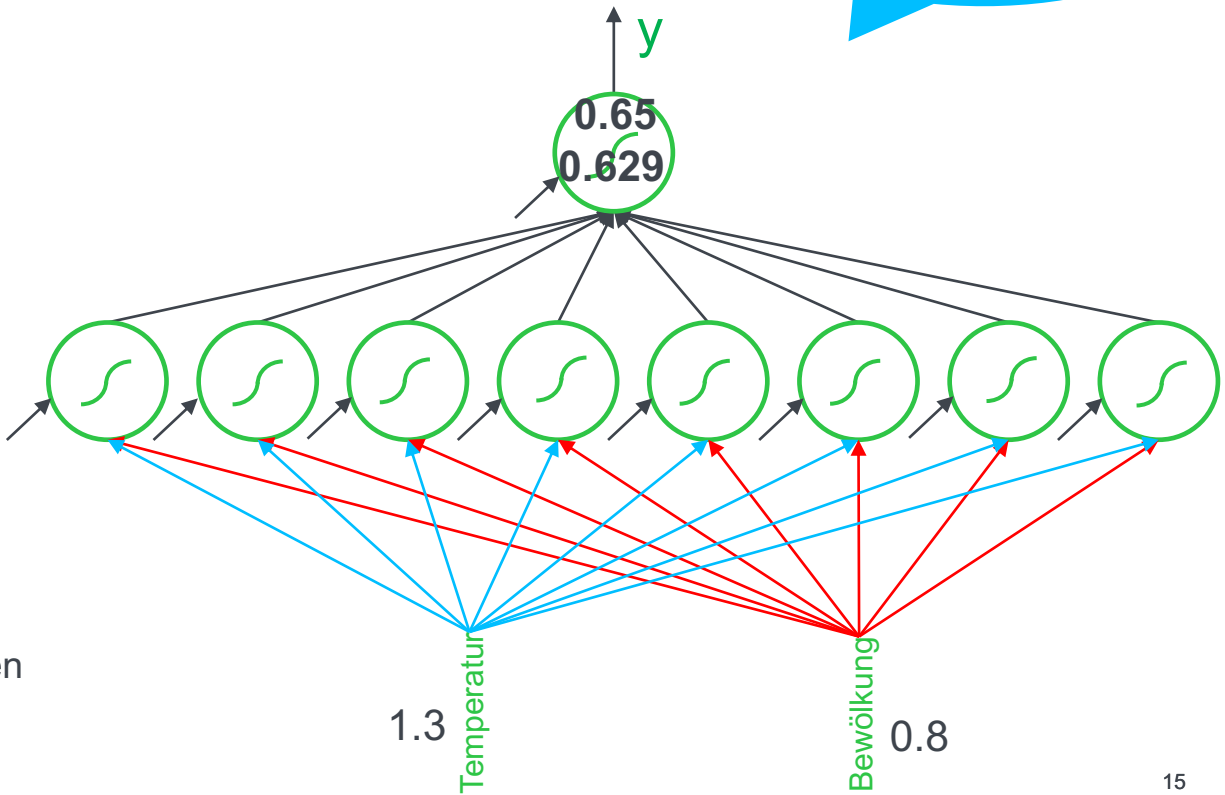
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung
  - Bias
- Sigmoid-Aktivierung der ersten Schicht (a_i^1)
- Logits der zweiten Schicht (z_i^2)
  - Inputs von Schicht 1
  - Bias



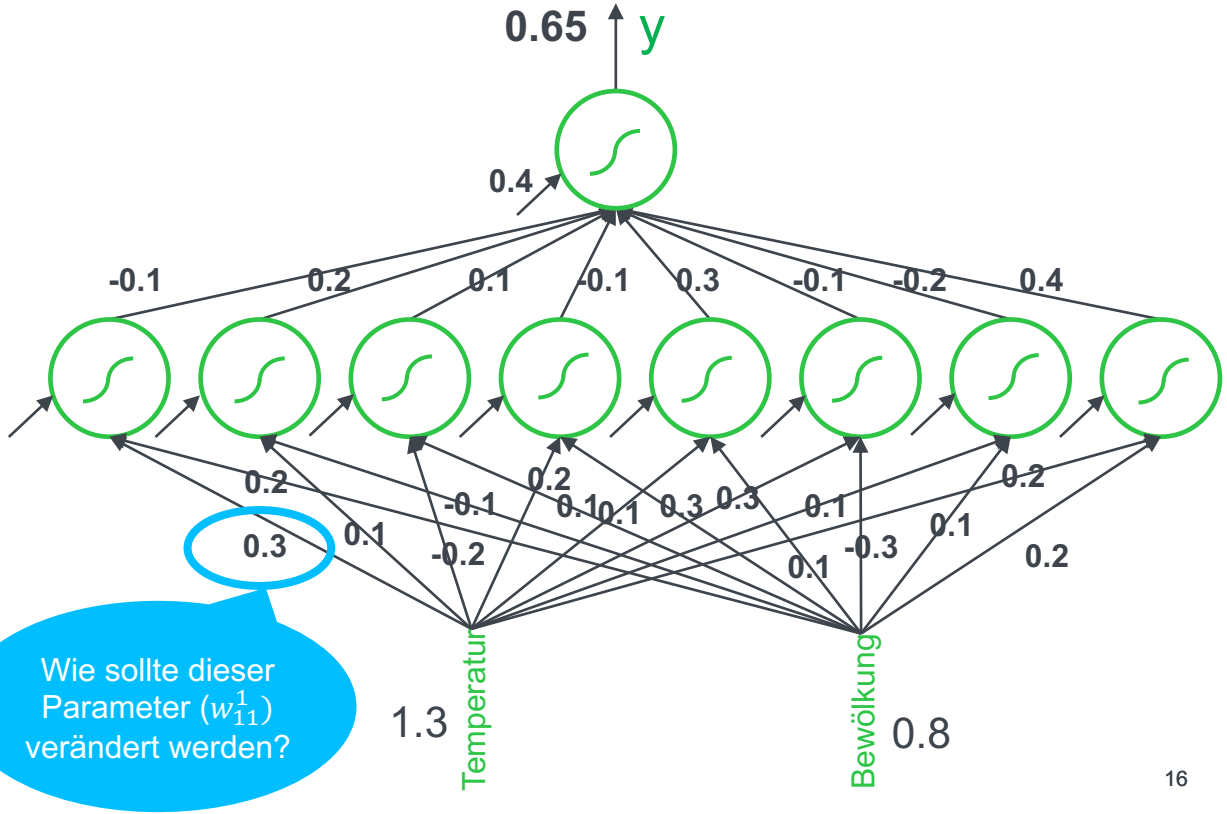
Forward Pass

- Logits der ersten Schicht (z_i^1)
  - Inputs von Temperatur
  - Inputs von Bewölkung
  - Bias
- Sigmoid-Aktivierung der ersten Schicht (a_i^1)
- Logits der zweiten Schicht (z_i^2)
  - Inputs von Schicht 1
  - Bias
- Sigmoid-Aktivierung der zweiten Schicht (a_i^1)



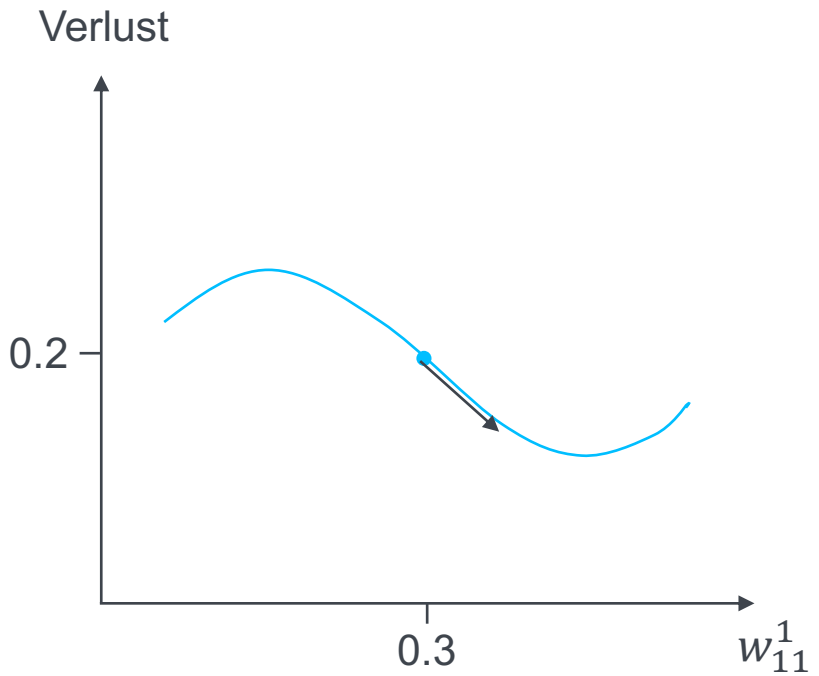
Evaluation

- Mit diesen Parametern:
  - Bei Input 1.3, 0.8
  - Ergebnis 0.65
- Abgleich mit korrektem Ergebnis:
  - Korrektes Ergebnis 0.45
  - „Verlust“ (Fehler): 0.2
- Gesucht:
  - Bessere Parameter
  - D.h. mit kleinerem Fehler



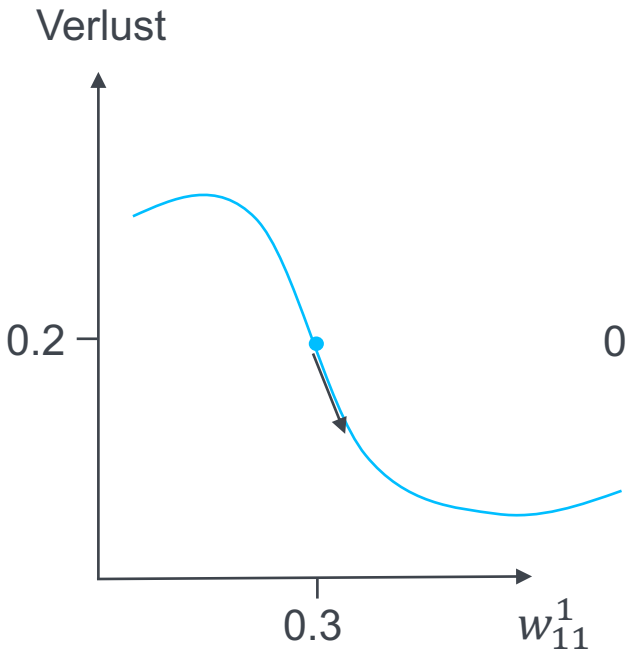
Wie verbessert man die Werte der Parameter?

- Beispiel: der Parameter von letzter Folie
- Verbindung zum ersten Neuron vom ersten Input in Schicht 1: w_{11}^1
- Der Verlust ist eine Funktion: wenn alle anderen Parameter konstant bleiben, ändert sich der Fehler abhängig von w_{11}^1
- Verändere Parameter w_{11}^1 so, dass der Verlust (also der Fehler!) kleiner wird
- Hier also: vergrößere w_{11}^1
- Um wie viel?

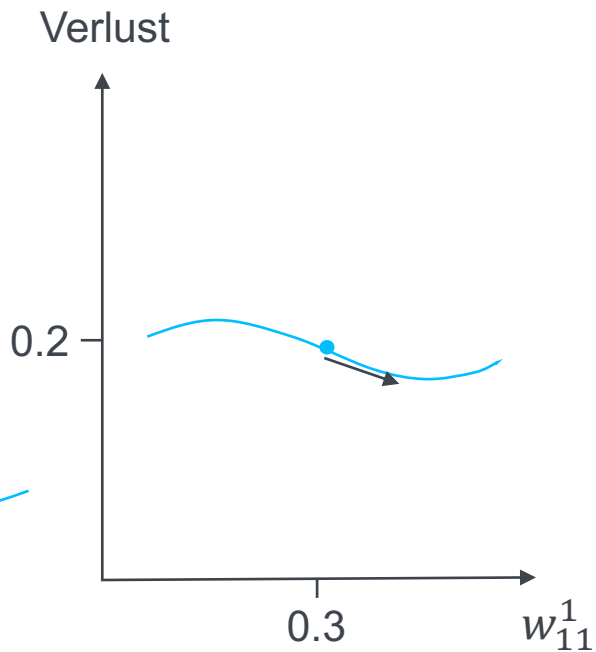


Wie bekommt man kleinere Fehler?

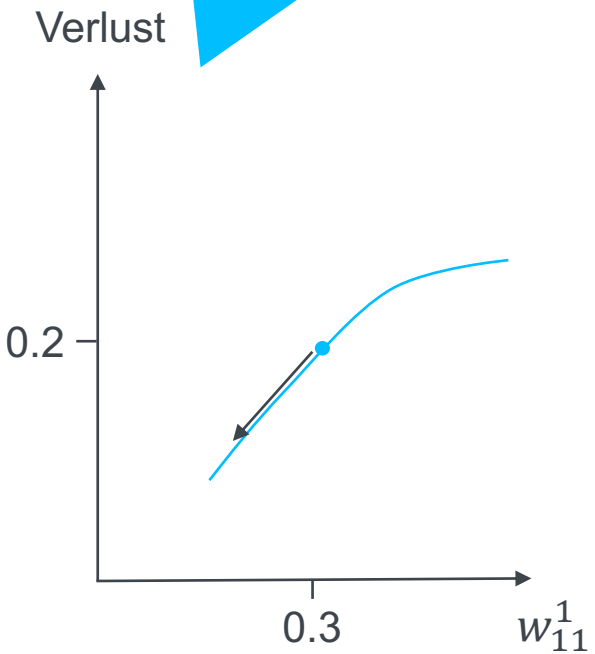
Im Maschinellen Lernen verbreiteter Begriff für Steigung: Gradient



Steigung stark negativ



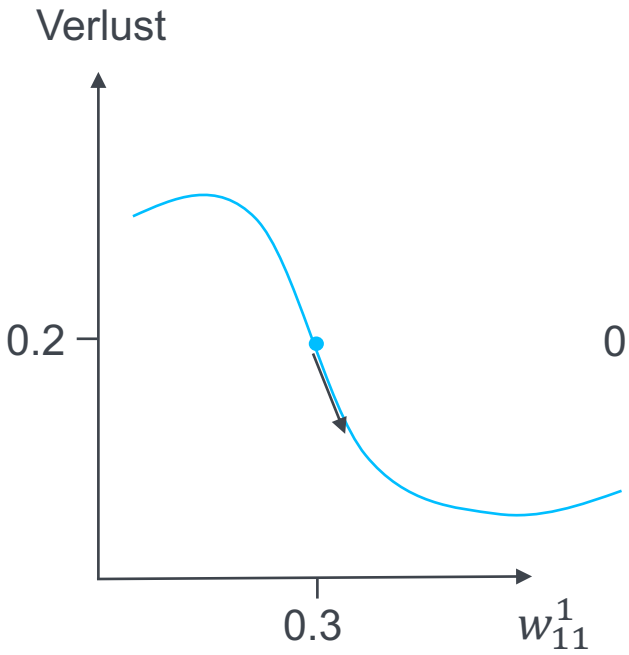
Steigung schwach negativ



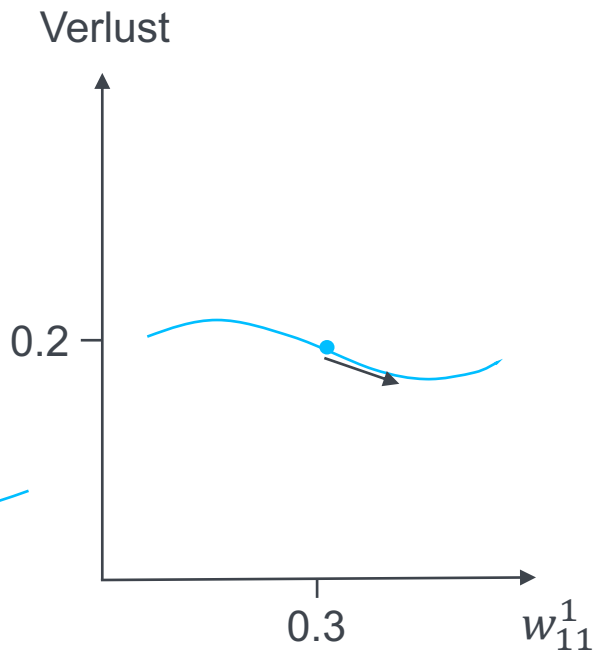
Steigung positiv

Gradientenabstiegsverfahren

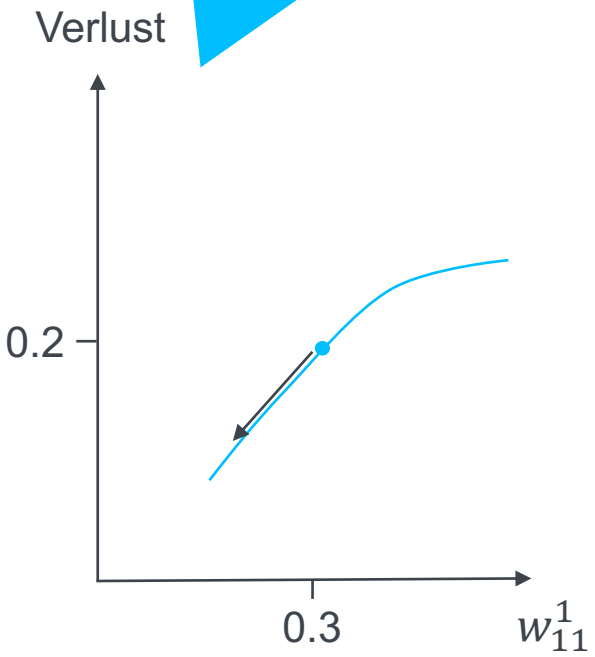
Gradient der Verlustfunktion für w_{11}^1 : ∂_{11}^1



Gradient stark negativ



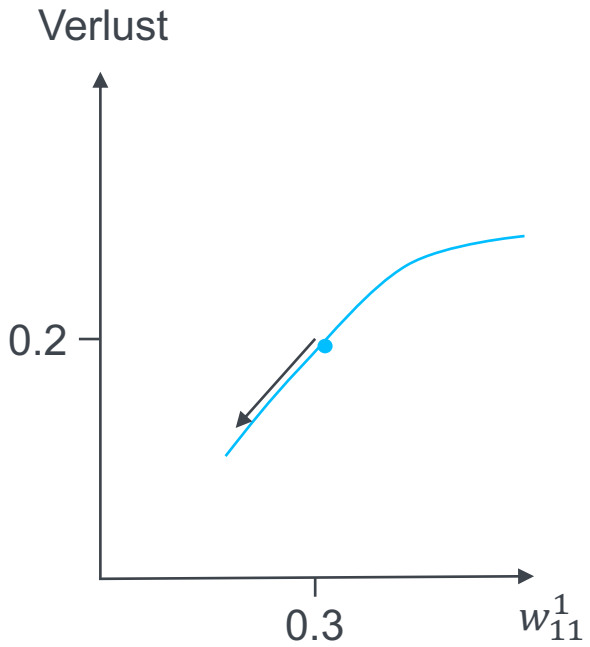
Gradient schwach negativ



Gradient positiv

Anpassung der Parameter beim Gradientenabstiegsverfahren

- ∂_{11}^{1} Gradient der Verlustfunktion für das erste Gewicht des ersten Neurons in Schicht 1
- ∂ spricht man „Delta“
- Korrektur für w_{11}^{1} also: $w_{11}^{1} = w_{11}^{1} - \partial_{11}^{1}$
  - Bewirkt Verringerung von w_{11}^{1}, falls Steigung positiv
  - Starke Erhöhung von w_{11}^{1}, falls Steigung stark negativ
  - Leichte Erhöhung von w_{11}^{1}, falls Steigung schwach negativ



Beim Gradientenabstiegsverfahren werden Werte optimiert, indem man sie gemäß der negativen Steigung der Verlustfunktion anpasst.

Dadurch ist eine Reduktion des Fehlers zu erwarten.

Der englische Begriff ist „Gradient Descent“.

Drag the Words: Gradient

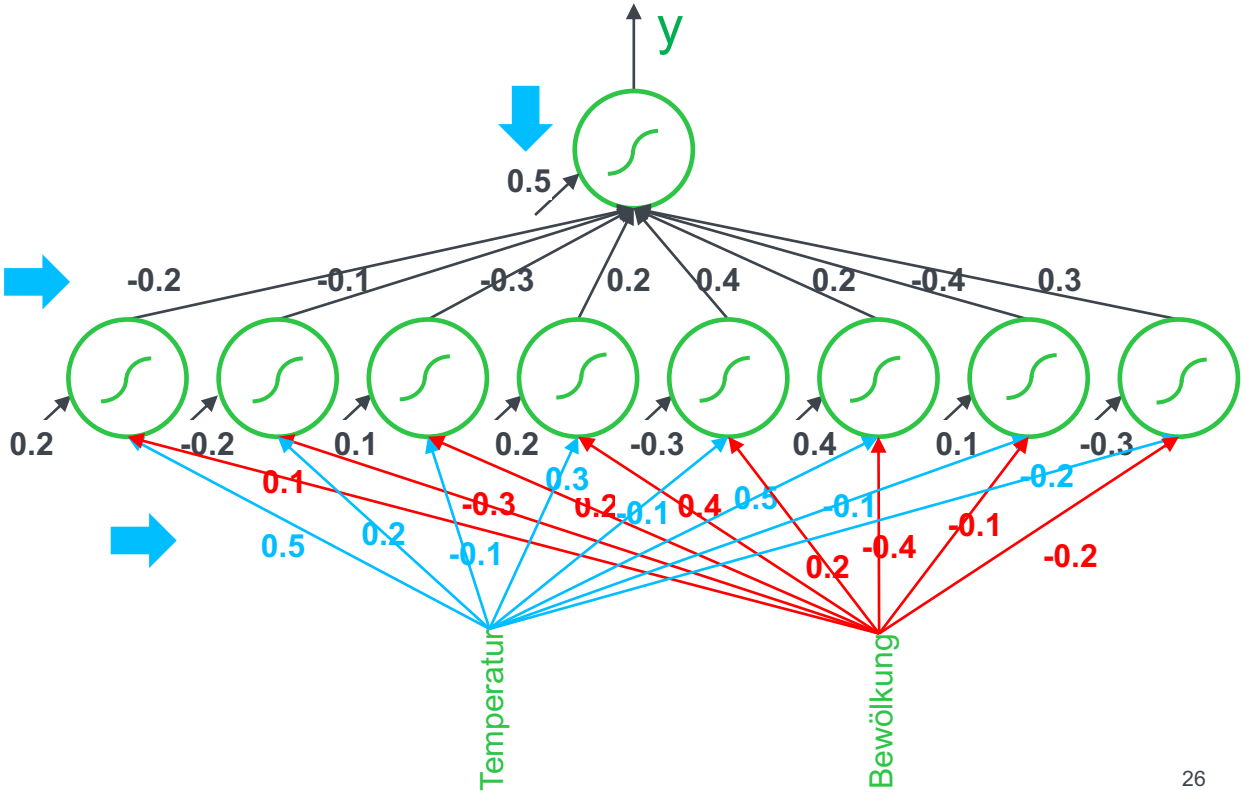
Drag and Drop: Die Steigung ist hier ...

Drag the Words: Gradient

Single Choice: Gradientenabstiegsverfahren

Beispiel: Backpropagation

- Backpropagation: Anpassung der Parameter gemäß Gradienten-abstiegsverfahren
- Beginnt in letzter Schicht
- Da für Berechnung der Gradienten die Gradienten der Schicht darüber benötigt werden



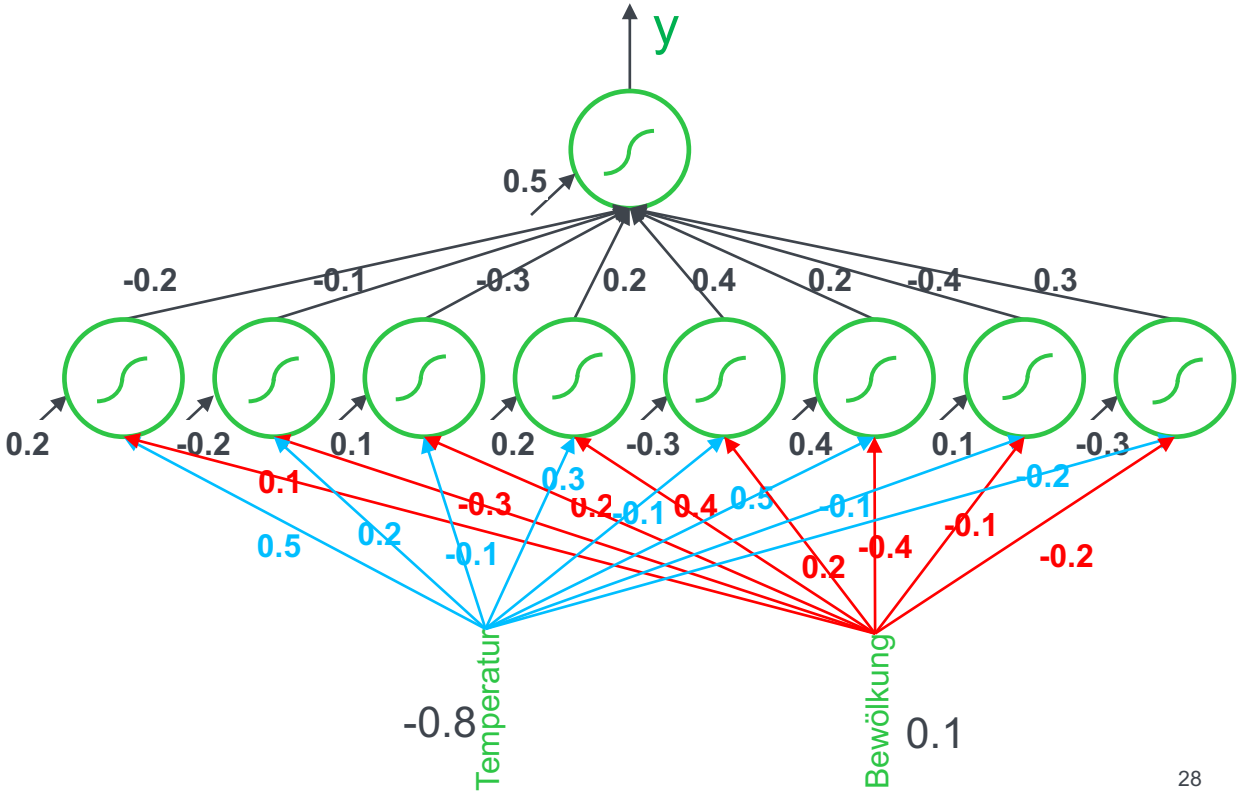
Backpropagation bezeichnet die Anpassung der Parameter beim Training neuronaler Netze.

Dabei beginnt man in der letzten Schicht, berechnet für die Parameter dort die Gradienten und passt die Parameter dann an.

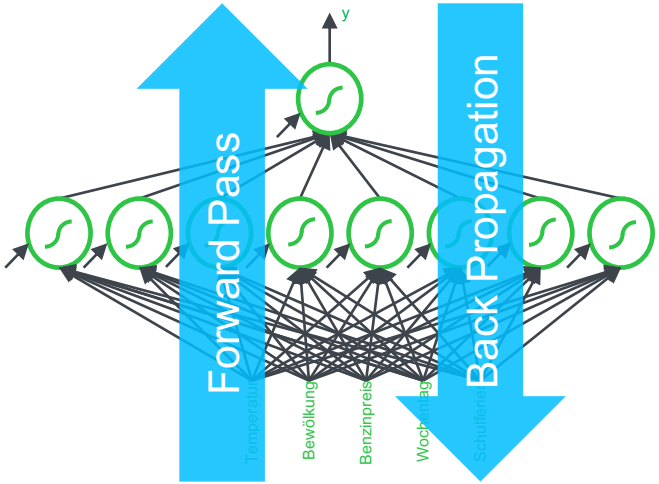
Anschließend geht man Schicht für Schicht zurück, berechnet mithilfe der bereits bekannten Gradienten die der vorhergehenden Schicht, und passt die Parameter dort ebenfalls an.

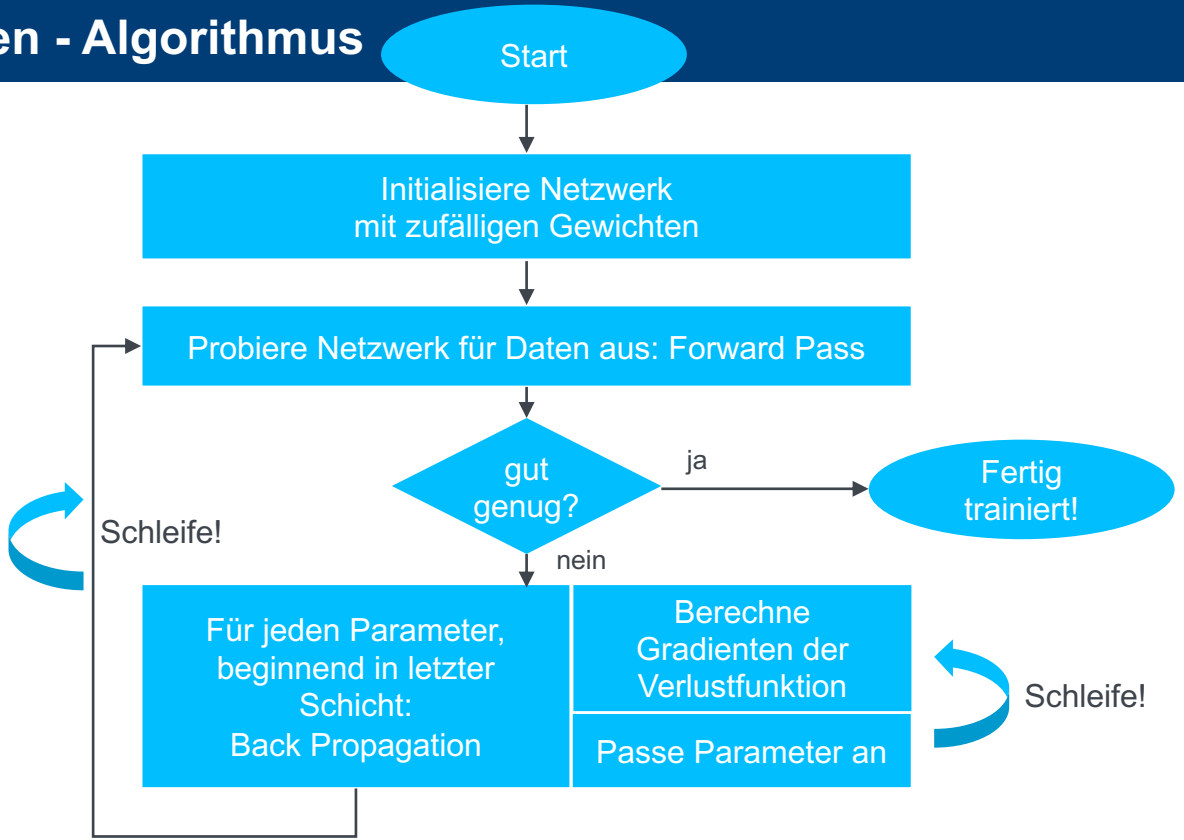
Neue Daten zum Testen

- Ausprobieren des neuen Netzes
- Mit weiteren Datenpunkten als Input
- D.h. Forward Pass mit den neuen Datenpunkten
- Anschließend Verlust berechnen



Training von neuronalen Netzen - Algorithmus





Bemerkungen zum Training

- Starke Vereinfachung
  - Einzelne Parameter so angepasst, als gäbe es keine weiteren Parameter:
  - Gradient der Verlustfunktion nur in Abhängigkeit des jeweiligen Parameters
- Aber: Falls Anpassung nicht gut war, hoffentlich Korrektur in der nächsten Iteration
- Was bedeutet “gut genug“?
  - Eigenes Video
- Das Training kann durch viele verschiedene Hyperparameter modifiziert werden, z.B.
  - Lernrate (wie stark soll der Gradient bei der Anpassung gewichtet werden?)
  - Batch Size (wie viele Datenpunkte werden für die Berechnung des Fehlers verwendet?)
  - Ebenfalls in diesem Video

Üben, üben, üben?

- Beim Training von Neuronalen Netzen sind häufig Hunderttausende von Iterationen nötig
- Rechenzeiten von mehreren Wochen sind keine Seltenheit, vor allem, wenn man Zugriff auf nur wenige GPUs hat
  - GPU: moderner Grafikprozessor, der für das Training von Neuronalen Netzen besonders geeignet ist

Drag the Words: Back Propagation

Multiple Choice: Training neuronaler Netze

INNOVET
KI B³
Künstliche Intelligenz in die berufliche Bildung bringen

Dr. Antje Schweitzer

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung
Institut für Software Engineering

IHK Industrie- und Handelskammer Reutlingen
Reutlingen | Tübingen | Zollernalb

IHK Region Stuttgart

IHK Industrie- und Handelskammer Karlsruhe

LMU Ludwig-Maximilians-Universität München

GEFÖRDERT VOM
INNOVET
Bundesministerium für Bildung und Forschung
bibb Bundesinstitut für Berufsbildung

Gefördert als InnoVET-Projekt aus Mitteln des Bundesministeriums für Bildung und Forschung.

Lizenzbestimmungen

“Training von Neuronalen Netzen" von Antje Schweitzer, KI B³ / Uni Stuttgart

Das Werk - mit Ausnahme der folgenden Elemente:

- Logos der Verbundpartner und des Förderprogramms
- im Quellenverzeichnis aufgeführte Medien

ist lizenziert unter:

 CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/deed.de)
(Namensnennung 4.0 International)

Quellenverzeichnis

Titelfoto: Stripe Media (https://unsplash.com/@stripemedia), “Hovering“, auf Unsplash (https://unsplash.com/photos/LdxOdbEsUO8), ist lizenziert unter Unsplash-Lizenz (https://unsplash.com/license). Bildausschnitt verändert.