Algorithmik zur Optimierung in neuronalen Netzwerken

Gradient Descent und Backpropagation

Tim Hilt

19. Mai 2020

Hochschule Esslingen — University of Applied Sciences

Gliederung

Supervised Learning

Machine Learning Workflow

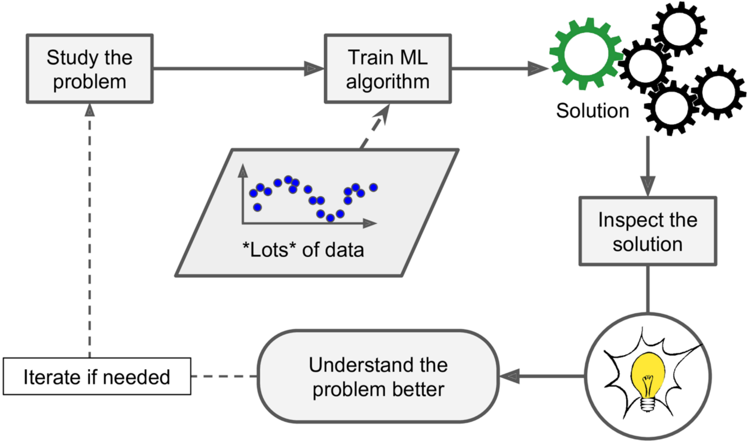


Abbildung 1: Machine Learning Workflow [1]

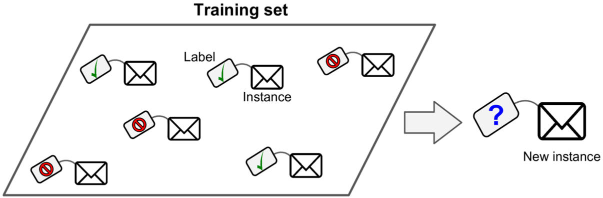


Abbildung 2: Struktur der Daten bei Supervised Learning [1]

Supervised Learning

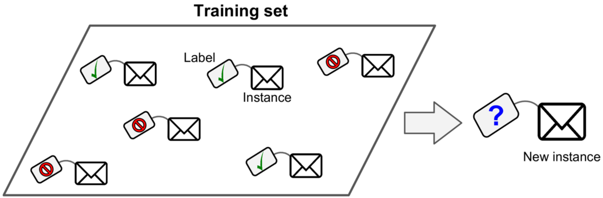
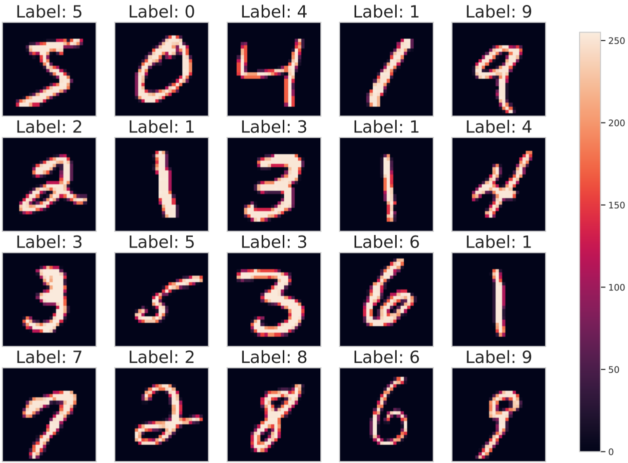


Abbildung 2: Struktur der Daten bei Supervised Learning [1]

Definition Supervised Learning

„In supervised learning, the dataset is the collection of labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. Each element $\mathbf{x}_i$ among N is called a feature vector. A feature vector is a vector in which each dimension $j = 1, \ldots, D$ contains a value that describes the example somehow [...]. The goal of a supervised learning algorithm is to use the dataset to produce a model, that takes a feature vector $\mathbf{x}$ as input and outputs information that allow deducing the label $\hat{y}$ for this feature vector." [2]

Beispiel: Datensatz für Supervised Learning



- Insgesamt 70000 Bilder
- Bildgröße: 28 × 28 Pixel
- Abgebildet: Handgeschriebene Ziffern von 0 bis 9
- Quelle: Yann LeCun et al [3]

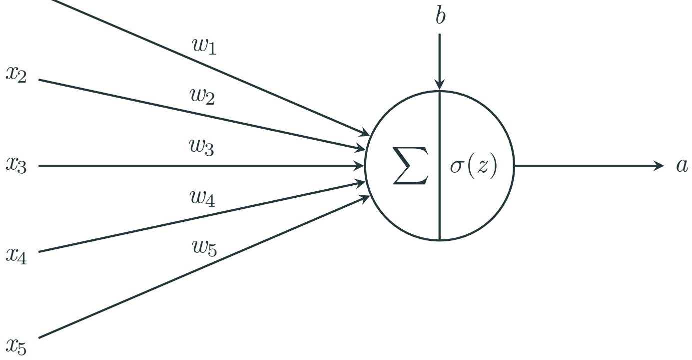
Beispiel: Datensatz für Supervised Learning

- Insgesamt 70000 Bilder
- Bildgröße: 28 × 28 Pixel
- Abgebildet: Kleidungsstücke
- Quelle: Zalando Research [4]

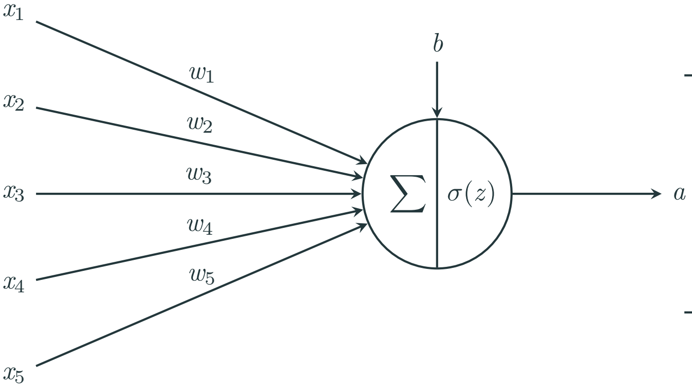
Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Künstliche Neuronale Netze

Künstliches Neuron

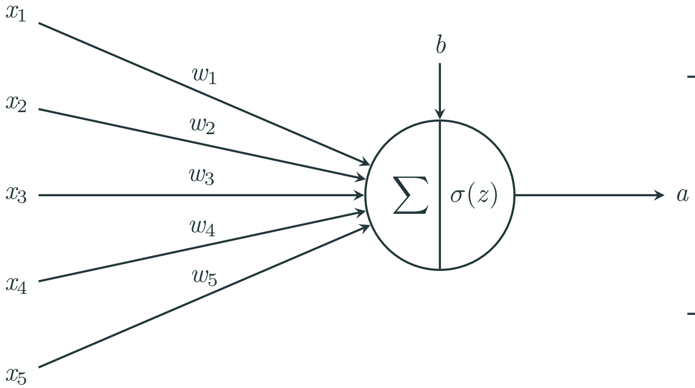


Künstliches Neuron



$\mathbf{x}$	Inputvektor
$\mathbf{w}$	Gewichtsvektor
b	Bias
z	Zwischensumme ($\sum$)
$\sigma(x)$	Aktivierungsfunktion
a	Aktivierung/ Output

Künstliches Neuron



$\mathbf{x}$	Inputvektor
$\mathbf{w}$	Gewichtsvektor
b	Bias
z	Zwischensumme ($\sum$)
$\sigma(x)$	Aktivierungsfunktion
a	Aktivierung/ Output

$$z = \sum_i x_i w_i + b = \mathbf{x}\mathbf{w} + b$$

$\Rightarrow$ z wird für spätere Parameteroptimierung benötigt

Aktivierungsfunktion $\sigma(x)$

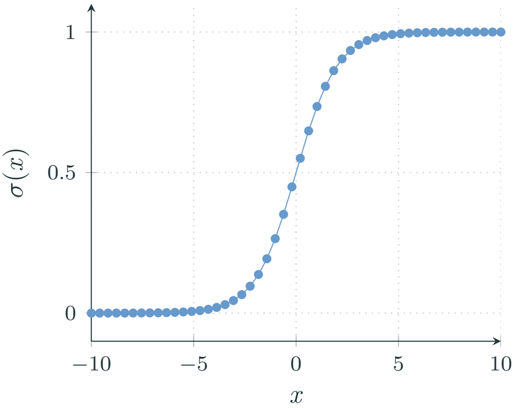
$\Rightarrow$ Es gibt eine Vielzahl verschiedener Aktivierungsfunktionen für unterschiedliche Problemstellungen, für uns soll jedoch lediglich die **Sigmoid-Funktion** relevant sein:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

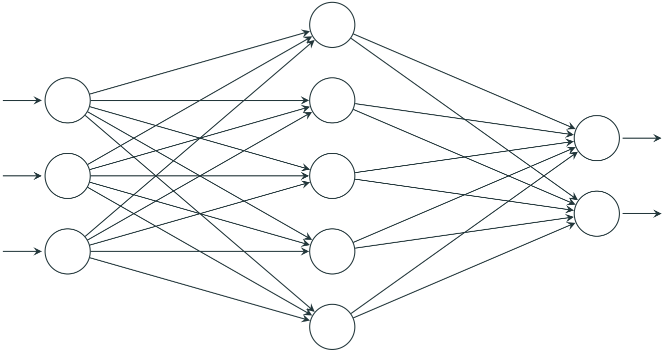
⇒ Es gibt eine Vielzahl verschiedener Aktivierungsfunktionen für unterschiedliche Problemstellungen, für uns soll jedoch lediglich die **Sigmoid-Funktion** relevant sein:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

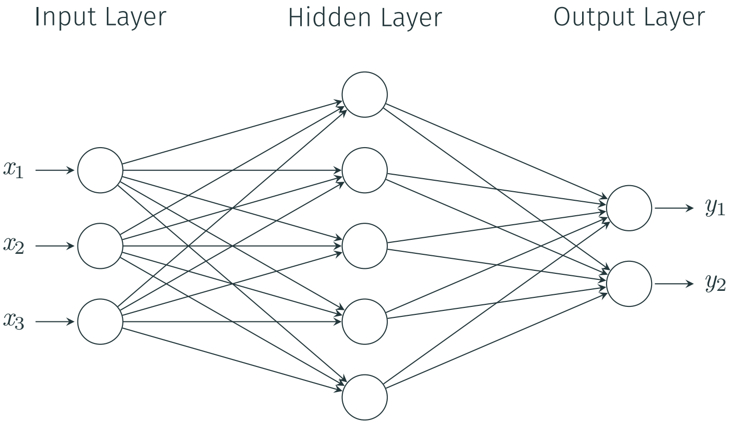
Sigmoid-Funktion



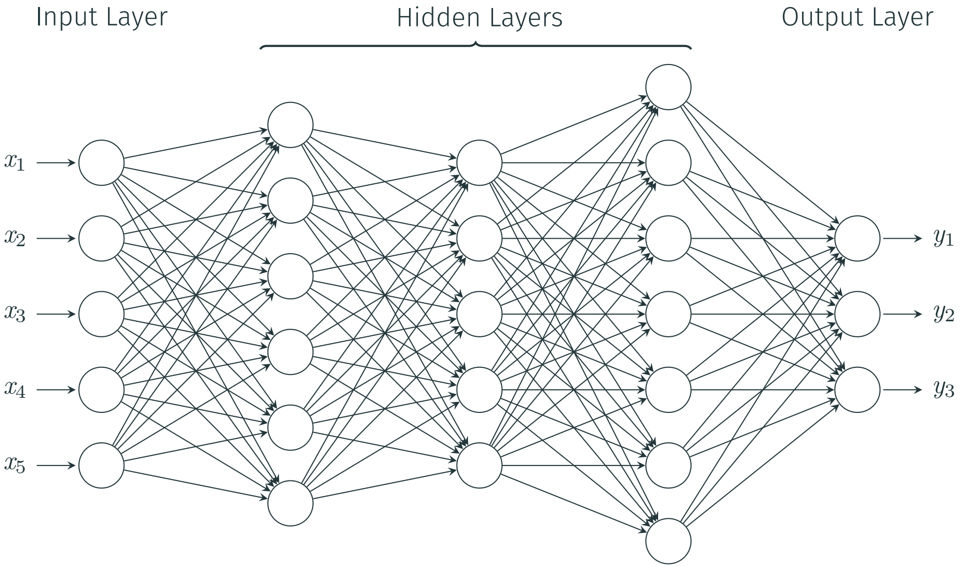
Architektur eines Neuronalen Netzwerks



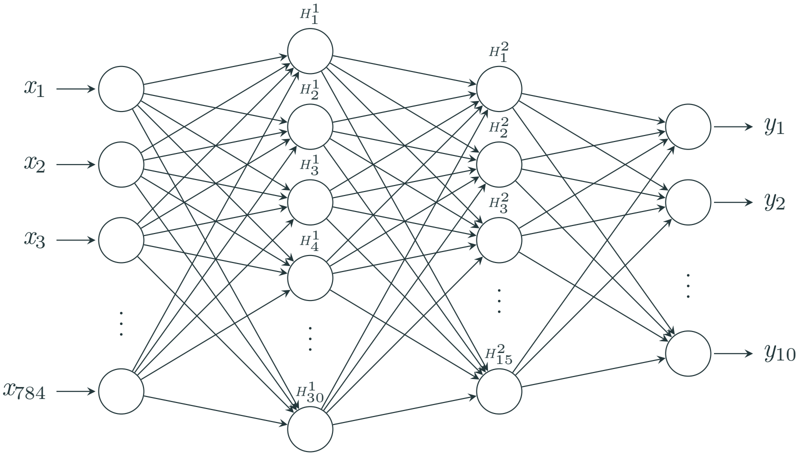
Architektur eines Neuronalen Netzwerks

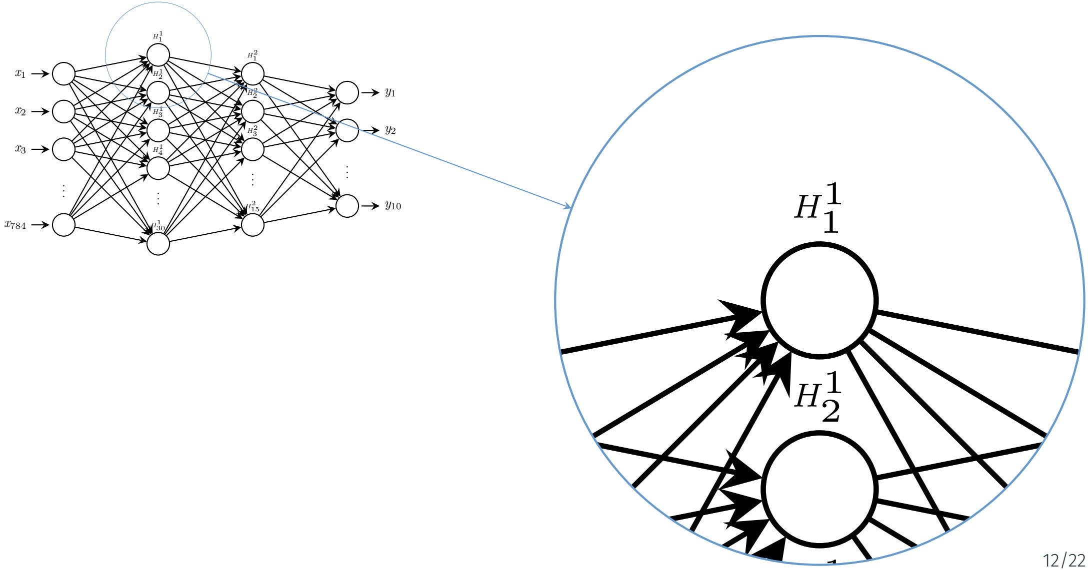


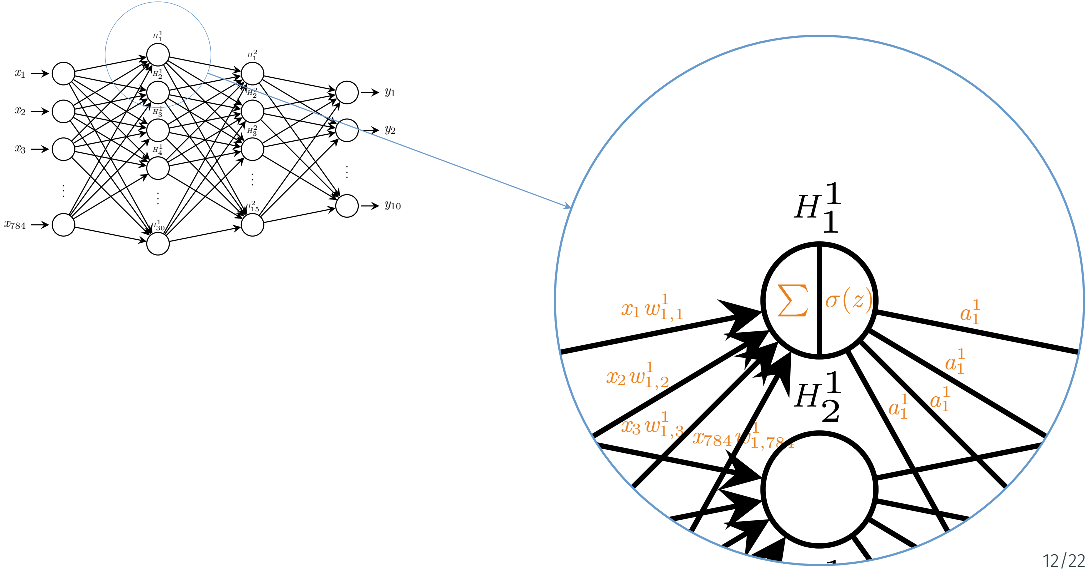
Deep Neural Network

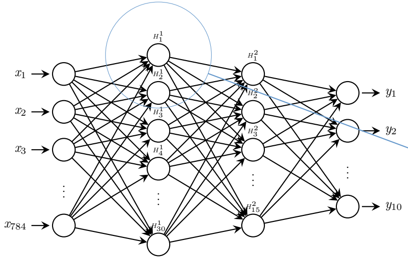


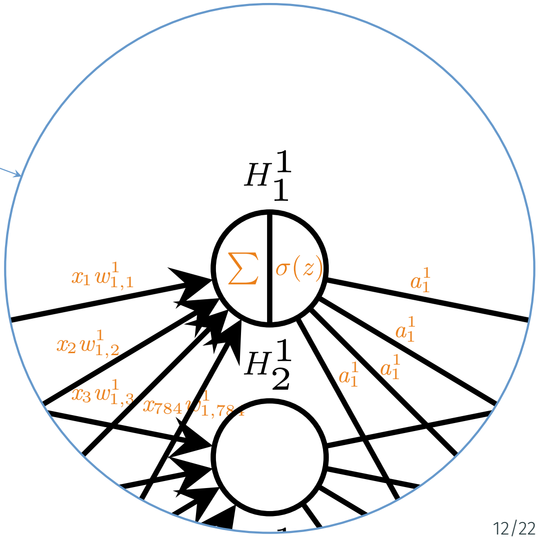
Target-Architektur zur Klassifikation von MNIST




x_1
x_2
x_3
x_{784}
H_1^1
H_2^1
H_3^1
H_4^1
H_{30}^1
H_1^2
H_2^2
H_3^2
H_{15}^2
y_1
y_2
y_{10}
H_1^1
H_2^1


x_1
x_2
x_3
x_{784}
H^1_1
H^1_2
H^1_3
H^1_4
H^1_{30}
H^2_1
H^2_2
H^2_3
H^2_{15}
y_1
y_2
y_{10}
$\sum$
$\sigma(z)$
$x_1\,w^1_{1,1}$
$x_2\,w^1_{1,2}$
$x_3\,w^1_{1,3}$
a^1_1





$$\mathbf{W^1} = \begin{bmatrix} w_{1,1}^1 & w_{2,1}^1 & \cdots & w_{30,1}^1 \\ w_{1,2}^1 & w_{2,2}^1 & \cdots & w_{30,2}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,784}^1 & w_{2,784}^1 & \cdots & w_{30,784}^1 \end{bmatrix} ; \; \mathbf{b^1} = \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{30}^1 \end{bmatrix}$$

Training

Loss-Funktion

- Dient zur Berechnung des Fehlers während dem Training
- Trainingsfehler soll minimiert werden
- $\Rightarrow$ wir suchen den Punkt, an dem die Ableitung der Loss-Funktion 0 wird, der Fehler also nicht mehr abnimmt
- Es gibt eine Vielzahl an Loss-Funktionen, wir betrachten hier die „**Mean Squared Error (MSE)**“:

$$C(w, b) = \frac{1}{2m} \sum_{x=1}^{m} (y(x) - \hat{y}(x))^2$$

Loss-Funktion

- Dient zur Berechnung des Fehlers während dem Training
- Trainingsfehler soll minimiert werden
- $\Rightarrow$ wir suchen den Punkt, an dem die Ableitung der Loss-Funktion 0 wird, der Fehler also nicht mehr abnimmt
- Es gibt eine Vielzahl an Loss-Funktionen, wir betrachten hier die „**Mean Squared Error (MSE)**“:

$$C(w,b) = \frac{1}{2m}\sum_{x=1}^{m}(y(x) - \hat{y}(x))^2$$

$C(w,b)$	Cost in Abhängigkeit von w und b
m	Anzahl der Trainingsinstanzen
$y(x)$	Gewünschter Output wenn x Input ist
$\hat{y}(x)$	Tatsächlicher Output des Netzwerkes

Gradient Descent

- Methode um die Weights w und Biases b zu optimieren
- Vorgehen:

Gradient Descent

- Methode um die Weights w und Biases b zu optimieren
- Vorgehen:
    1. Finde die Änderungsrate des Fehlers in Abhängigkeit von den Weights und Biases ($\partial C/\partial w; \partial C/\partial b$)

Gradient Descent

- Methode um die Weights w und Biases b zu optimieren
- Vorgehen:
  1. Finde die Änderungsrate des Fehlers in Abhängigkeit von den Weights und Biases ($\partial C/\partial w; \partial C/\partial b$)
  2. Multipliziere die Änderungsrate mit der Lernrate η

Gradient Descent

- Methode um die Weights w und Biases b zu optimieren
- Vorgehen:
  1. Finde die Änderungsrate des Fehlers in Abhängigkeit von den Weights und Biases ($\partial C / \partial w; \partial C / \partial b$)
  2. Multipliziere die Änderungsrate mit der Lernrate η
  3. Ziehe das Produkt aus Änderungsrate und Lernrate von den aktuellen Parametern ab

Gradient Descent

- Methode um die Weights w und Biases b zu optimieren
- Vorgehen:
  1. Finde die Änderungsrate des Fehlers in Abhängigkeit von den Weights und Biases ($\partial C/\partial w; \partial C/\partial b$)
  2. Multipliziere die Änderungsrate mit der Lernrate η
  3. Ziehe das Produkt aus Änderungsrate und Lernrate von den aktuellen Parametern ab
  4. Aktualisiere die alten Parameter durch das Ergebnis des letzten Schrittes

$$w_{k+1} = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_{k+1} = b_k - \eta \frac{\partial C}{\partial b_k}$$

Backpropagation

Problem

Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}; \frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

Backpropagation

Problem

Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}$; $\frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

$\Rightarrow$ **Idee:** Divide and conquer; Problem in kleinere, handhabbare Probleme zerlegen

Problem

Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}; \frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

$\Rightarrow$ **Idee:** Divide and conquer; Problem in kleinere, handhabbare Probleme zerlegen

$$f(a,b,c,d) = ((a+b)\cdot(c+d))^2$$

Backpropagation

Problem

Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}; \frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

$\Rightarrow$ **Idee:** Divide and conquer; Problem in kleinere, handhabbare Probleme zerlegen

$$f(a, b, c, d) = ((a+b) \cdot (c+d))^2$$
$$g(a, b) = a + b$$

Backpropagation

Problem

Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}; \frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

$\Rightarrow$ **Idee:** Divide and conquer; Problem in kleinere, handhabbare Probleme zerlegen

$$
\begin{aligned}
f(a, b, c, d) &= ((a + b) \cdot (c + d))^2 \\
g(a, b) &= a + b \\
h(c, d) &= c + d
\end{aligned}
$$

Problem

Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}; \frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

$\Rightarrow$ **Idee:** Divide and conquer; Problem in kleinere, handhabbare Probleme zerlegen

$$
\begin{aligned}
f(a,b,c,d) &= ((a+b)\cdot(c+d))^2 \\
g(a,b) &= a+b \\
h(c,d) &= c+d \\
i(g,h) &= g\cdot h
\end{aligned}
$$

Backpropagation

Problem

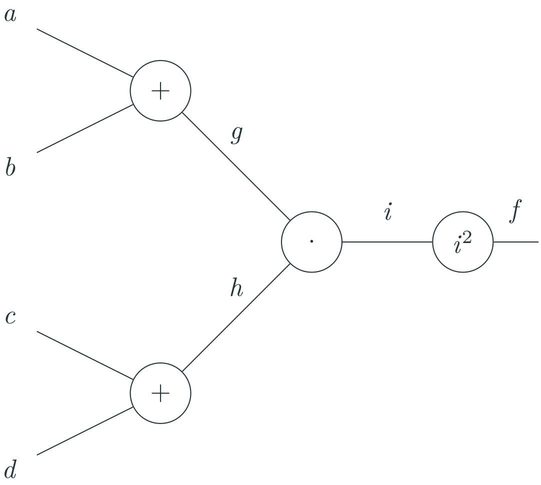
Wie finde ich die Änderungsraten $\frac{\partial C}{\partial w}; \frac{\partial C}{\partial b}$, die ich für Gradient Descent benötige?

$\Rightarrow$ **Idee:** Divide and conquer; Problem in kleinere, handhabbare Probleme zerlegen

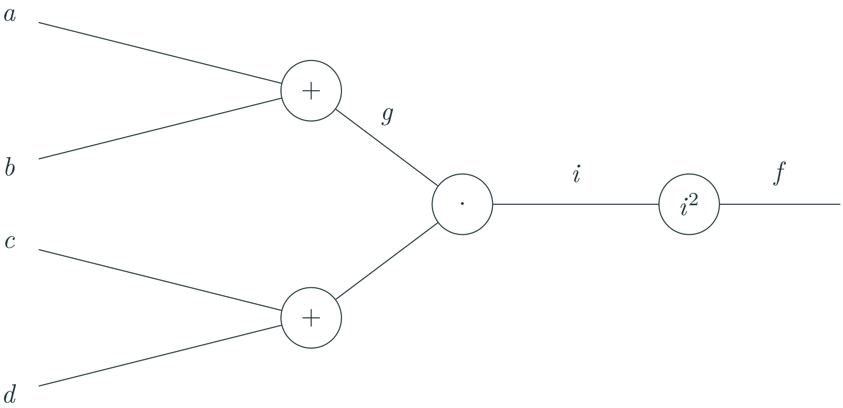
$$
\begin{aligned}
f(a,b,c,d) &= ((a+b)\cdot(c+d))^2 \\
g(a,b) &= a+b \\
h(c,d) &= c+d \\
i(g,h) &= g\cdot h \\
f(i) &= i^2
\end{aligned}
$$

Backpropagation

$$f(a, b, c, d) = ((a + b) \cdot (c + d))^2$$
$$g(a, b) = a + b$$
$$h(c, d) = c + d$$
$$i(g, h) = g \cdot h$$
$$f(i) = i^2$$

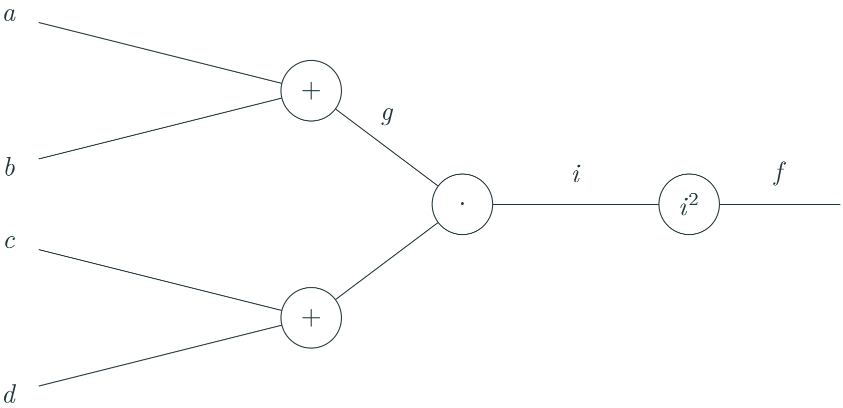


Backpropagation



Frage: $\frac{\partial f}{\partial a}$?

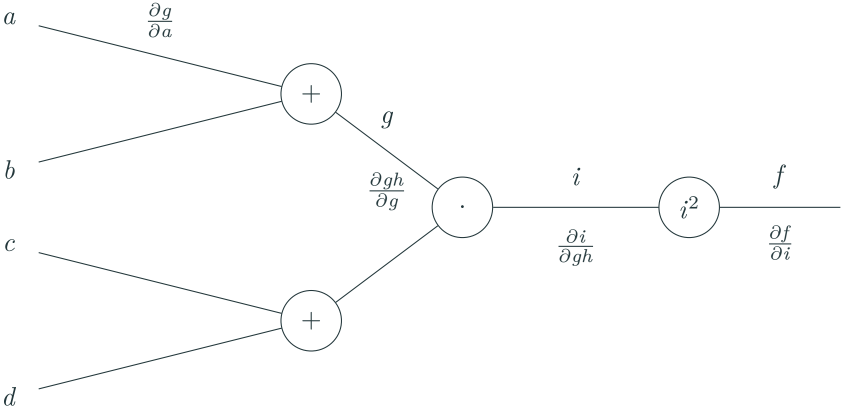
Backpropagation



Frage: $\frac{\partial f}{\partial a}$?

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial i} \cdot \frac{\partial i}{\partial gh} \cdot \frac{\partial gh}{\partial g} \cdot \frac{\partial g}{\partial a}$$

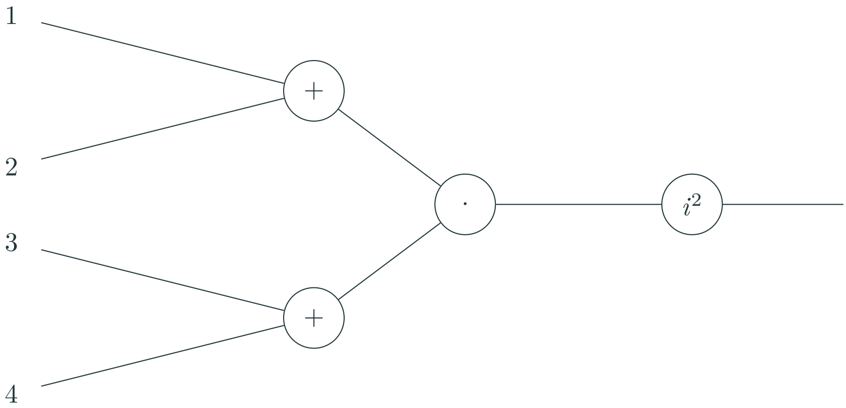
Backpropagation



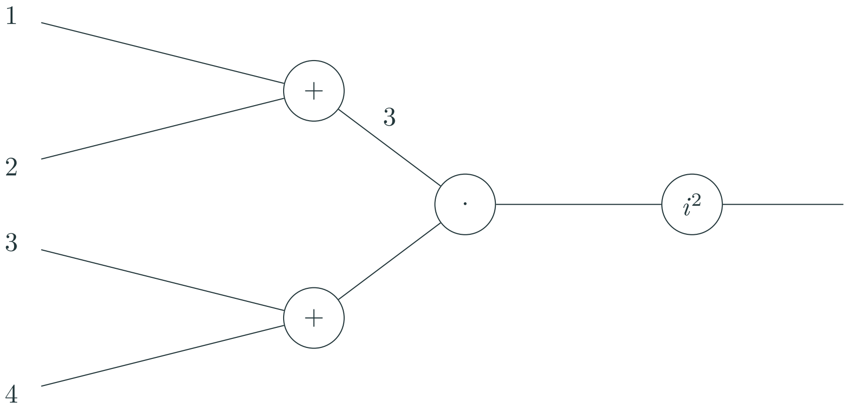
Frage: $\frac{\partial f}{\partial a}$?

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial i} \cdot \frac{\partial i}{\partial gh} \cdot \frac{\partial gh}{\partial g} \cdot \frac{\partial g}{\partial a}$$

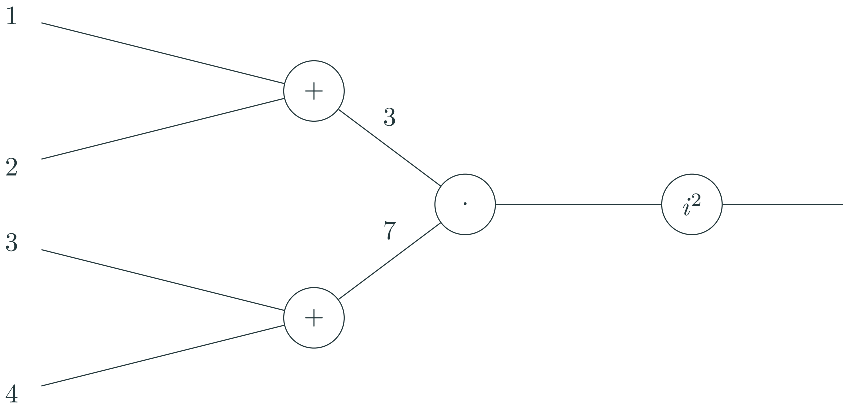
Beispiel



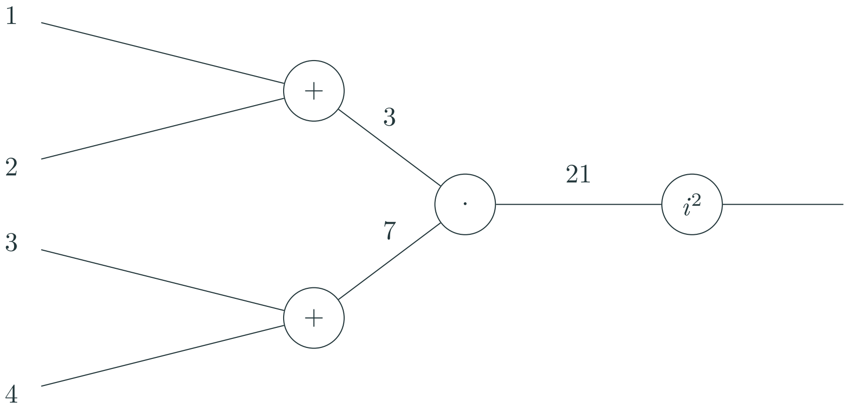
Beispiel



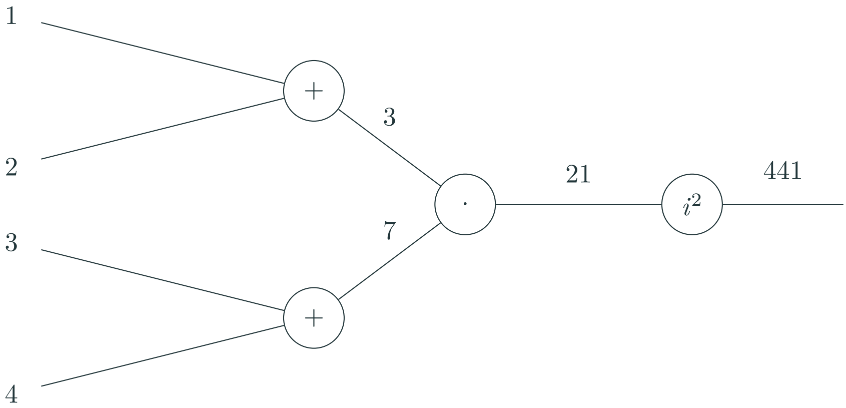
Beispiel



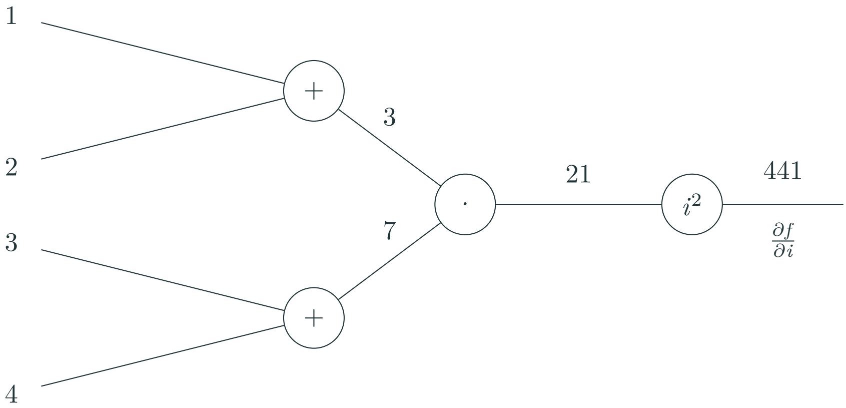
Beispiel



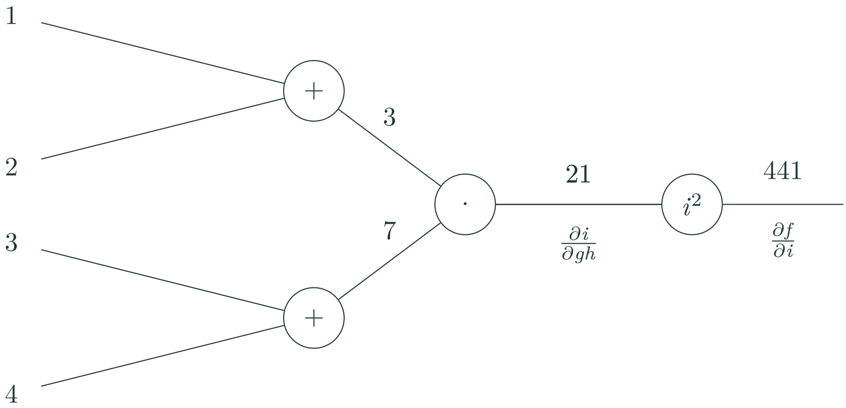
Beispiel



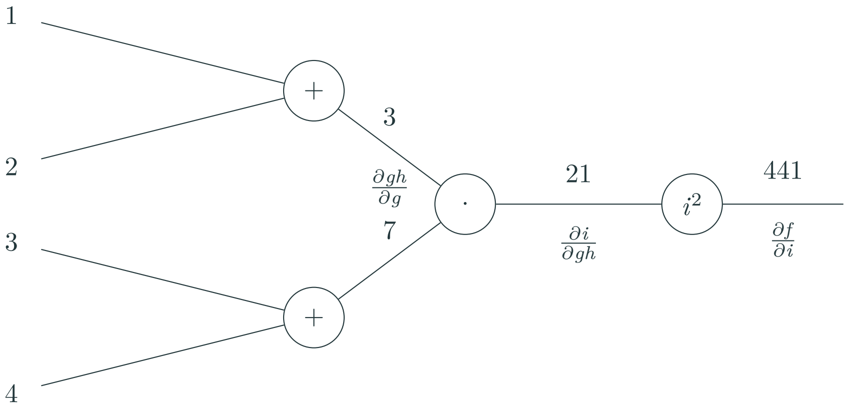
Beispiel



Beispiel

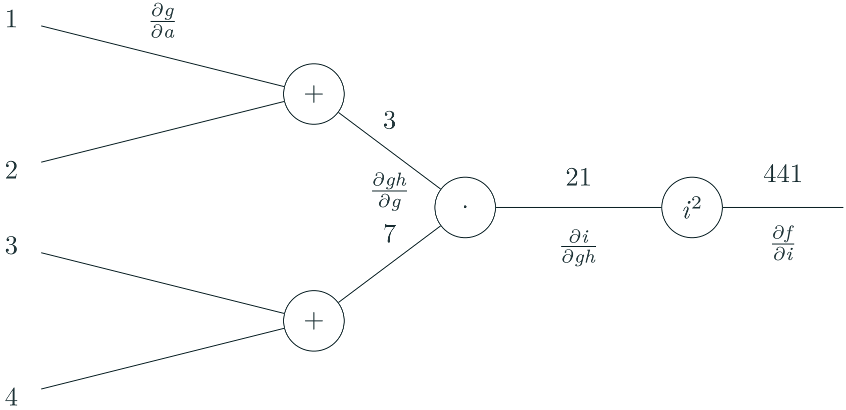

1
2
3
4
+
+
3
7
·
21
i^2
441
$\frac{\partial i}{\partial gh}$
$\frac{\partial f}{\partial i}$

Beispiel



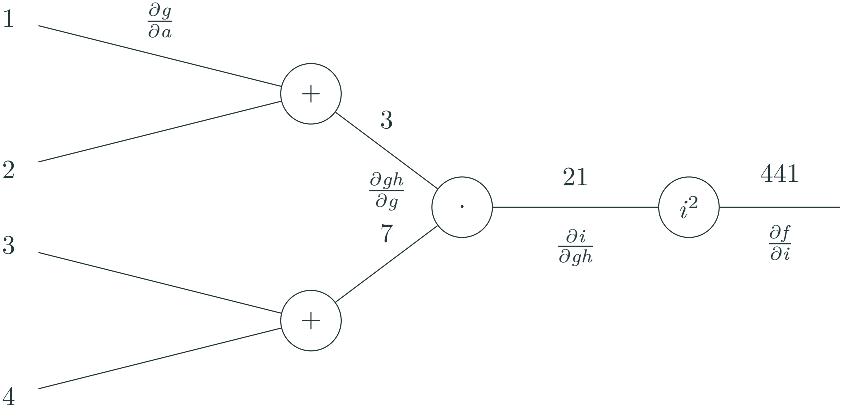
1
2
3
4
+
+
3
$\frac{\partial gh}{\partial g}$
7
$\cdot$
21
$\frac{\partial i}{\partial gh}$
i^2
441
$\frac{\partial f}{\partial i}$

Beispiel



1
$\frac{\partial g}{\partial a}$
2
3
4
+
+
3
$\frac{\partial gh}{\partial g}$
7
·
21
$\frac{\partial i}{\partial gh}$
i^2
441
$\frac{\partial f}{\partial i}$

Beispiel



$$\frac{\partial f}{\partial i} = 2 \cdot 21 = 42; \quad \frac{\partial i}{\partial gh} = 1; \quad \frac{\partial gh}{g} = h = 7; \quad \frac{\partial g}{\partial a} = 1 \quad \Rightarrow 7 \cdot 42 = 294$$

Umsetzung in Keras

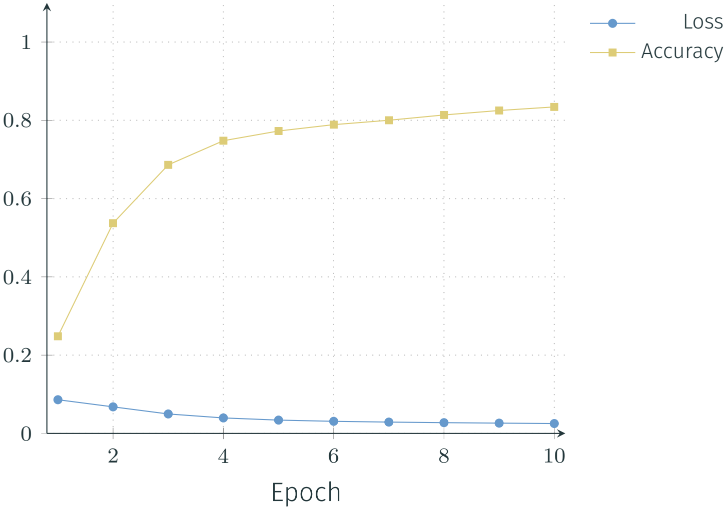
Zuvor beschriebene Architektur

```
from tensorflow import keras
...
model = keras.models.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(30, activation='sigmoid'),
  keras.layers.Dense(15, activation='sigmoid'),
  keras.layers.Dense(10, activation='sigmoid'),
])

model.compile(loss='mse',
              optimizer=keras.optimizers.SGD(learning_rate=.8),
              metrics=['accuracy'])

history = model.fit(X_train, y_train_cat, epochs=10,
                    validation_data=(X_valid, y_valid_cat))
```

Verlauf während des Trainings



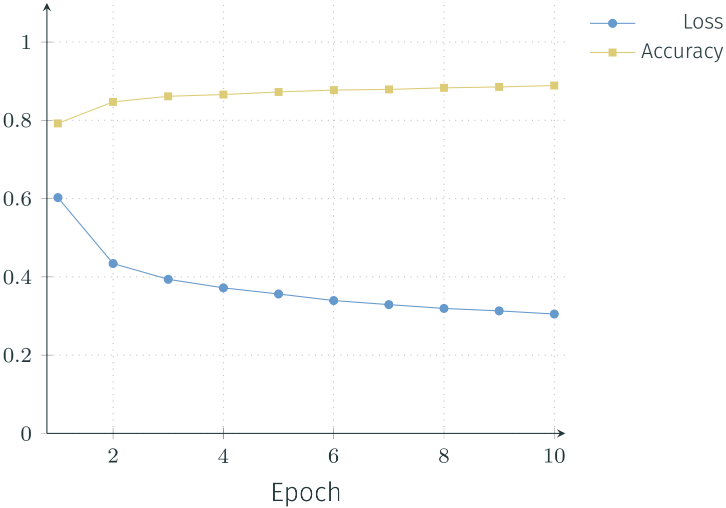
Optimierte Architektur

- Vorteil: Schnellere Konvergenz
- Verwendung von optimierter Cost-, Activation- und Gradient-Descent-Funktion

```
model = keras.models.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(30, activation='relu'),
  keras.layers.Dense(15, activation='relu'),
  keras.layers.Dense(10, activation='softmax'),
])

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])
```

Verlauf während des Trainings



Fragen?

Literaturverzeichnis

GÉRON, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

BURKOV, Andriy. *The hundred-page machine learning book*. Andriy Burkov Quebec City, Can., 2019.

LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, Jg. 86, Nr. 11, S. 2278–2324.

XIAO, Han; RASUL, Kashif; VOLLGRAF, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. 2017.