Architektur Neuronaler Netze für Generative KI

Grundlagen Neuronaler Netze 1

Logistische Regression, Gradient Descent, etc.

Hochschule Worms • Fachbereich Informatik

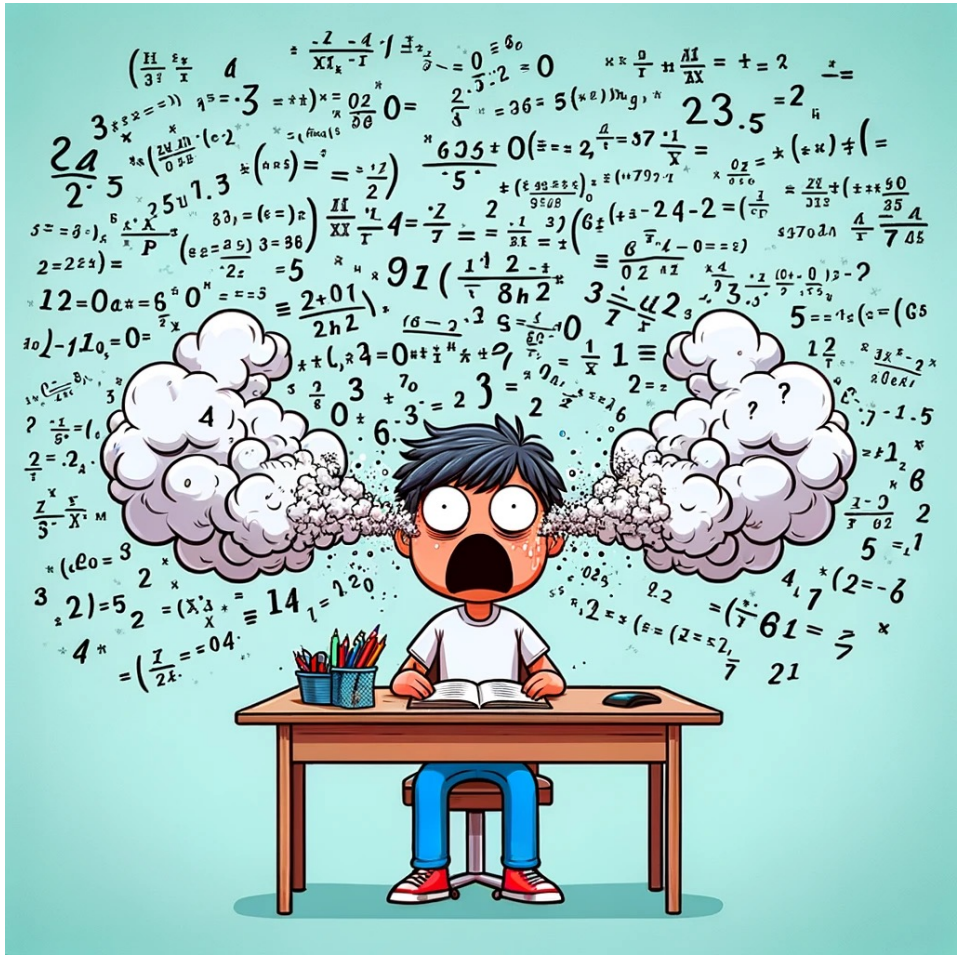
Prof. Dr. Stephan Kurpjuweit

Einführung

Vorbemerkung

Heute geht es um die Grundlagen neuronaler Netze. Dabei wird es etwas mathematischer. Diese Aspekte einmal zu durchdenken hilft uns aber zu verstehen, was in den neuronalen Netzen intern passiert.

In den nächsten Veranstaltungen soll die Praxis aber nicht zu kurz kommen.



Quelle: DALL-E

Grundidee

- Modelle lernen Parameter.
- Eine Fehlerfunktion misst die Differenz zwischen dem vom Modell vorhergesagten Wert und dem tatsächlichen Wert.
- Beim Trainieren des Modells berechnen wir den Fehler mit Hilfe der Fehlerfunktion und passen dann die Parameter des Modells an, um den Fehler zu minimieren. Diesen Vorgang wiederholen wir so lange, bis wir den Fehler auf ein akzeptables Niveau reduziert haben. Gradientenabstieg (Gradient Descent) ist einer der wichtigsten Algorithmen hierzu.

Themen

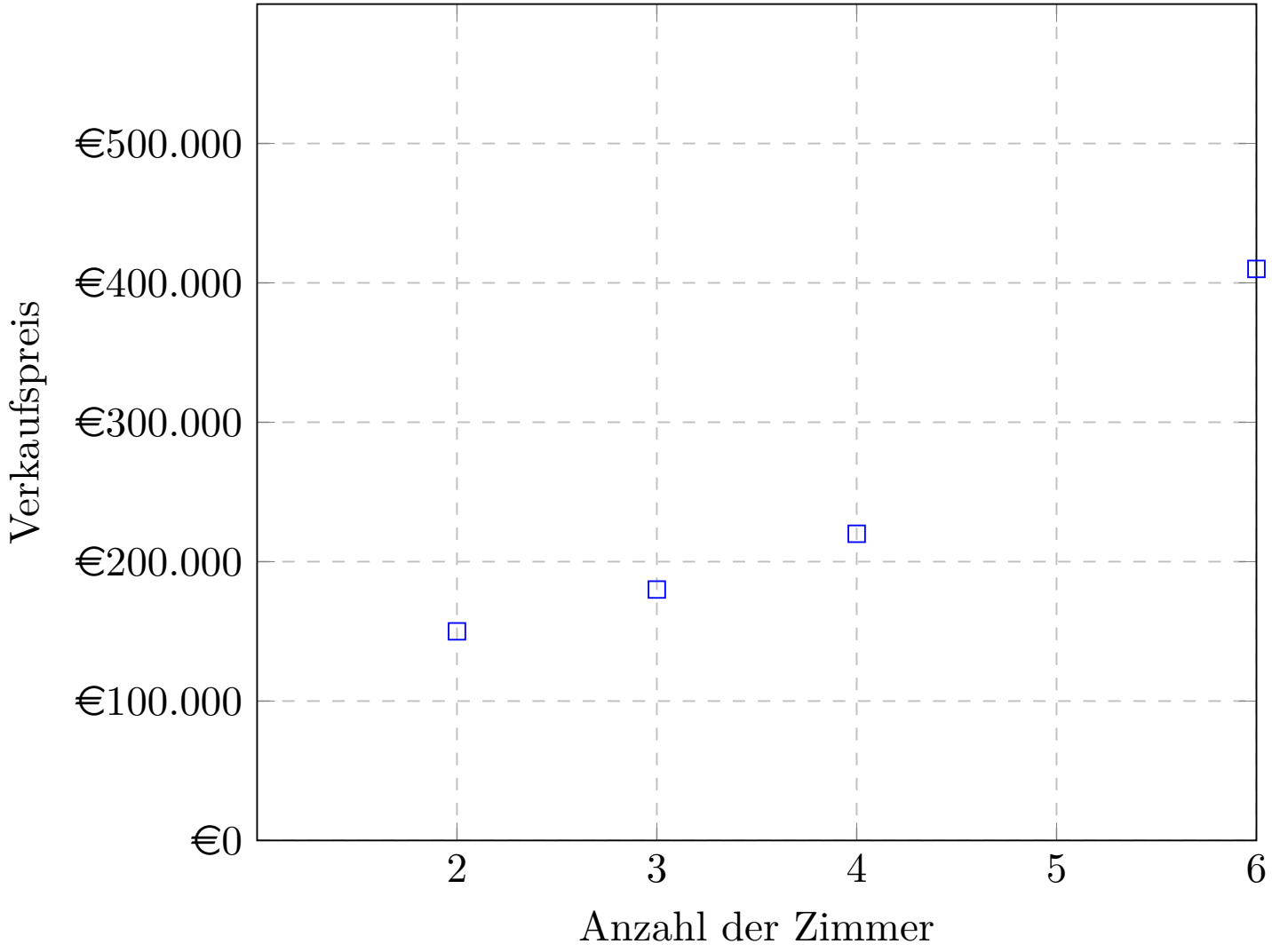
- Lineare Regression
- Binäre Klassifikation
- Logistische Regression
- Kostenfunktion und Gradient Descent
- Wiederholung: Ableitung einer Funktion
- Berechnung der Ableitung im Berechnungsgraph
- Die Ableitung im Gradient Decsent
- Kostenfunktion bei der logistischen Regression
- Gradient Descent bei der logistischen Regression

Lineare Regression

Beispiel: Immobilienpreise

Anzahl der Zimmer	Verkaufspreis
2	€150.000
3	€180.000
4	€220.000
6	€410.000

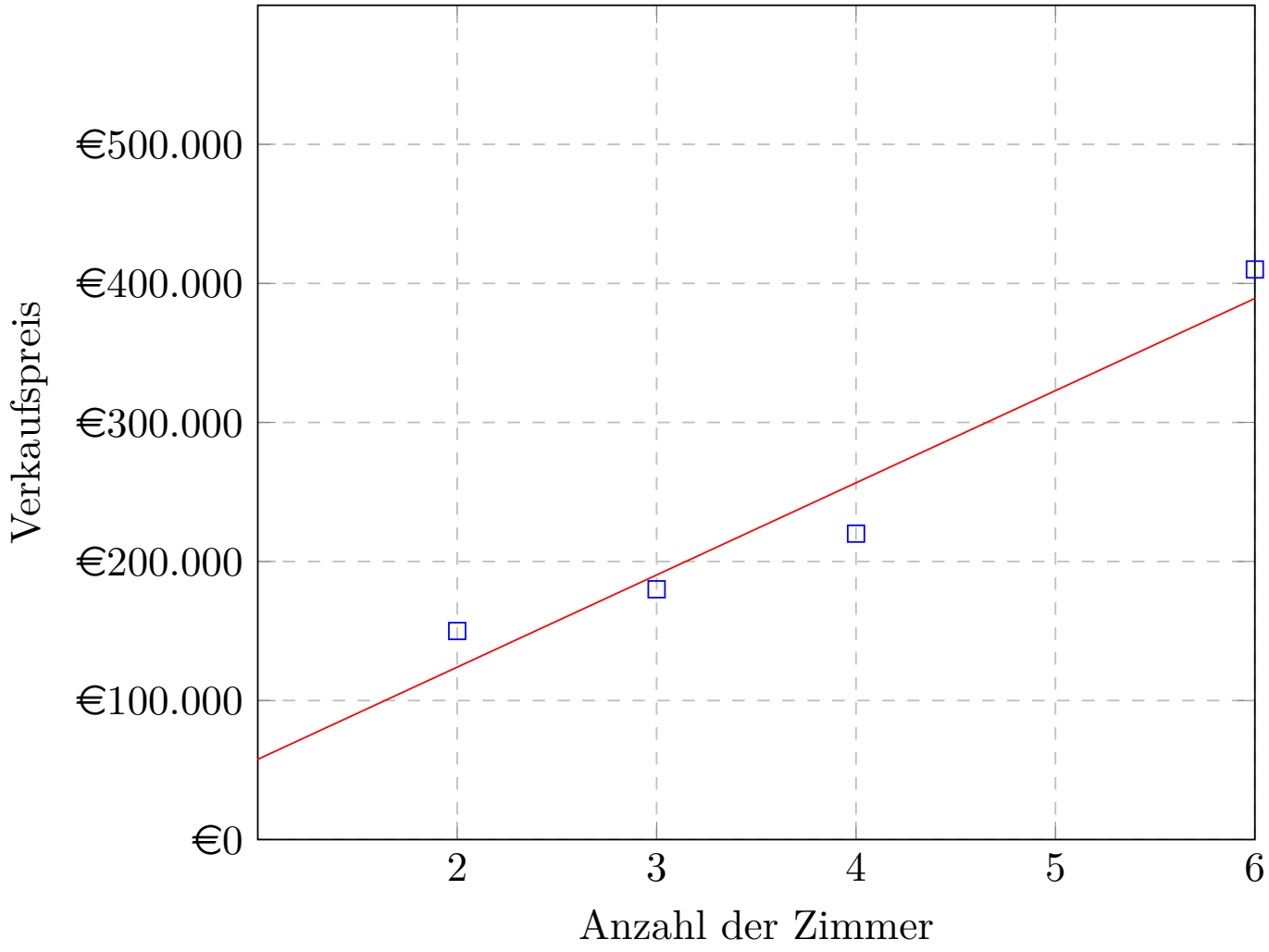
Ziel: Berechnen einer Geraden, die den Zusammenhang zwischen der Anzahl der Zimmer und dem Verkaufspreis beschreibt.



Geradengleichung

$$y = w_1 x + b$$

Es geht also darum,
die Parameter
w_1 und b
zu berechnen.



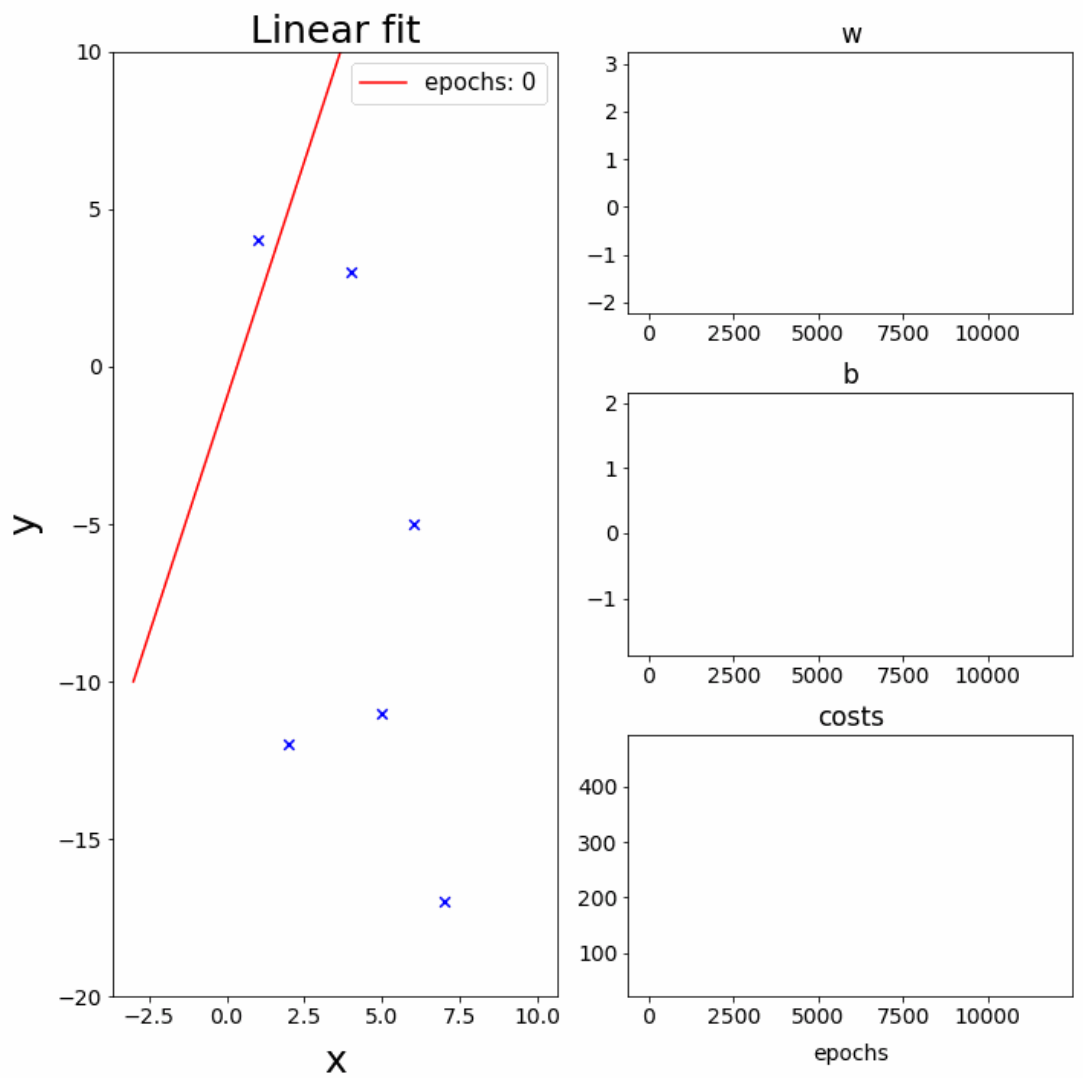
Grundidee der Berechnung

- Die Berechnung erfolgt iterativ, ausgehend von zufälligen Werten.
- Der Abstand der Punkte von der Linie kann zur Definition einer Kostenfunktion genutzt werden (z.B. Residual Sum of Squares, RSS):

$$\mathrm{RSS}(w_1, b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_1 \cdot x_i + b))^2$$

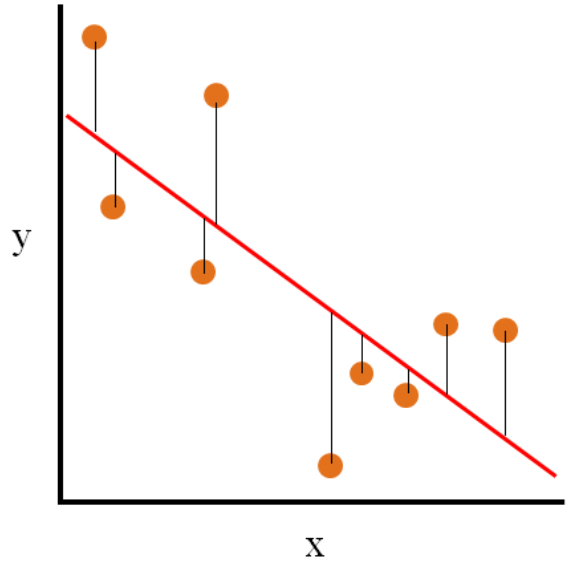
- Durch die Quadratur der Abstände können sich positive und negative Abweichungen nicht ausgleichen.
- Die Parameter werden so verändert, dass der Fehlerwert nach und nach minimiert wird.



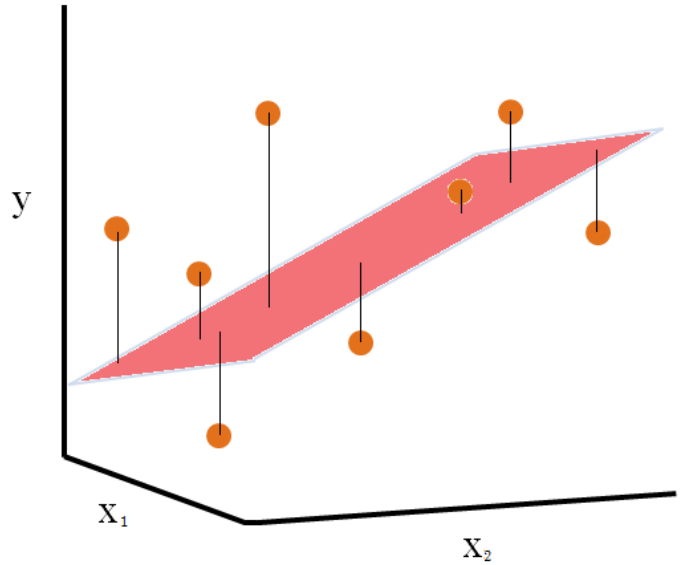
https://towardsdatascience.com/gradient-descent-animation-1-simple-linear-regression-e49315b24672

mit mehreren Eingangsvariablen (Features)

Einfache lineare Regression



Multiple lineare Regression



https://www.keboola.com/blog/linear-regression-machine-learning

Allgemeine Gleichung

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b$$

Binäre Klassifikation

Binäre Klassifikation

Die binäre Klassifizierung ist die Aufgabe, die Elemente einer Menge auf der Grundlage einer Klassifizierungsregel in eine von zwei Gruppen (jeweils Klasse genannt) einzuordnen.
Das Ergebnis einer binären Klassifizierung nennt man binäre Klassifikation.

Beispiel:

$$\begin{cases} 1 & \text{für} \quad Katze \\ 0 & \text{für} \quad keine\ Katze \end{cases}$$

x

y

Bild als Eingabe

$$x = \begin{pmatrix} 67 \\ 167 \\ \vdots \\ 71 \\ 250 \\ 145 \\ \vdots \\ 170 \\ 15 \\ 10 \\ \vdots \\ 37 \end{pmatrix}$$

Beispiel: 700 Pixel x 800 Pixel x 3 Schichten

$\Rightarrow$ Eingabegröße $n_x = 1.680.000$ (Dimensionen des Eingangsvektors)

Mathematische Notation

Größe eines Eingangsvektors: n_x

Trainingsbeispiel: $(x, y) \in \mathbb{R}^{n_x}, y \in \{0, 1\}$
Datensatz mit m Trainingsbeispielen: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

der Eingangsvektor des i-ten Trainingsbeispiels: $x^{(i)} \in \mathbb{R}^{n_x}$
Eingangsmatrix (mit allen Eingangsvektoren des Datensatzes): $X \in \mathbb{R}^{n_x \times m}$
$X = [x^{(1)}, x^{(2)}, \ldots, x^{(m)}]$

die Ausgabe (das Label) des i-ten Trainingsbeispiels: $y^{(i)} \in \{0, 1\}$
Ausgabematrix (Label-Matrix) (mit allen Labeln des Datensatzes): $Y \in \{0, 1\}^{1 \times m}$
$Y = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]$

Logistische Regression

Was ist eine logistische Regression?

- Bei der Regressionsanalyse handelt es sich um statistische Analyseverfahren, welche die Beziehungen zwischen einer abhängigen und einer oder mehreren unabhängigen Variablen modellieren.
- Die logistische Regression ist eine spezielle Regressionsanalyse und wird verwendet, wenn die abhängige Variable nominalskaliert ist, d.h. ihre Werte Klassen darstellen, die sich nicht in eine natürliche Reihenfolge bringen lassen.

Beispiele:

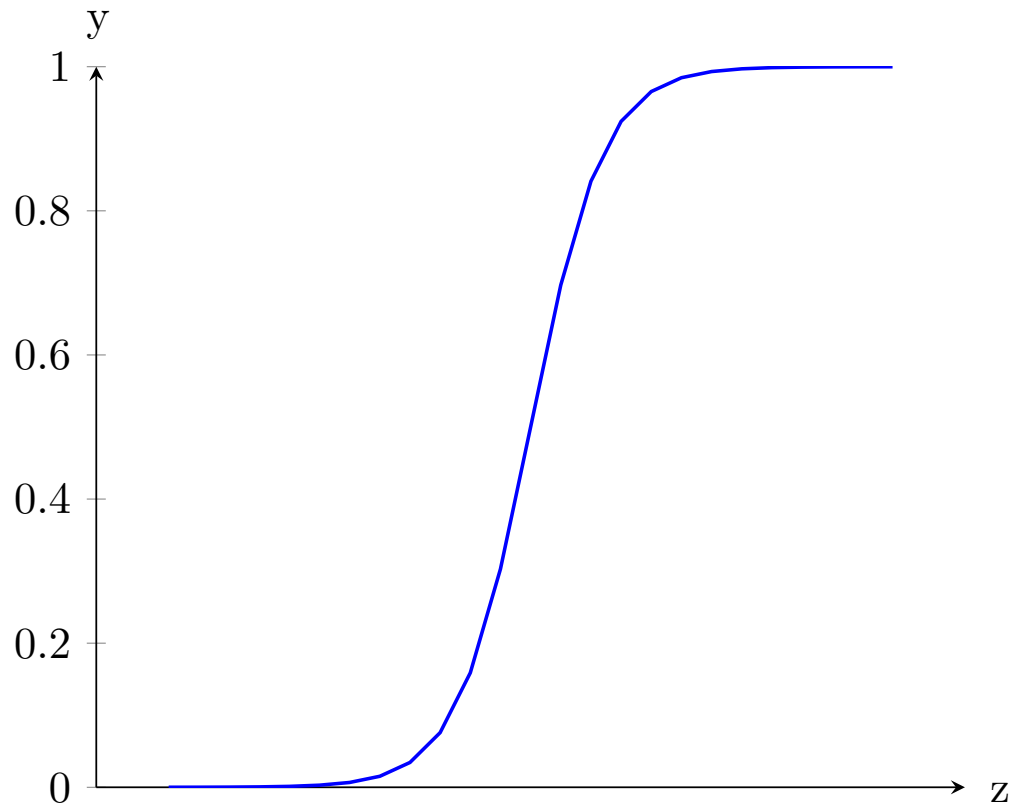
- Welche Produkte wird ein Kunde am ehesten kaufen?
  (oder: Mit welcher Wahrscheinlichkeit wird ein Kunde mit bestimmten Eigenschaften welches Produkt kaufen?)
- Welche Personen sind für eine Krankheit anfällig?
  (oder: Mit welcher Wahrscheinlichkeit wird eine Person mit bestimmten Eigenschaften die Krankheit bekommen?)
- Welche Partei wird von einer Person gewählt?
  (oder: Mit welcher Wahrscheinlichkeit entscheidet sich eine Person mit bestimmten Eigenschaften für welche Partei?)

Weitere Informationen: https://datatab.de/tutorial/logistische-regression

Logistische Funktion

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

kleines Sigma



- Die logistische Funktion ist eine spezielle Sigmoid-Funktion*. Sie bildet eine reelle Zahl auf den Werteberich zwischen 0 und 1 ab.
- Für große Zahlen nähert sie sich der 1 an, für kleine Zahlen der 0.

* So wird sie manchmal auch genannt, vor allem in englischen Quellen.

Logistische Regression - Mathematisch

Gegeben: $x \in \mathbb{R}^{n_x}$
Eingangsvektor, z.B. ein Bild

Berechnet werden soll: $\hat{y} = P(y = 1|x), 0 \leq \hat{y} \leq 1$
z.B. die Wahrscheinlichkeit, dass eine Katze auf dem Bild ist

Hierzu müssen die folgenden **Parameter** bestimmt werden: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

Wie die Parameter bestimmt werden, sehen wir später. Die Ausgabe wird mit Hilfe dieser Parameter wie folgt aus dem Eingangsvektor berechnet:

$$\hat{y} = \sigma(w^T x + b) = \sigma(w_1 x_1 + w_2 x_2 + \cdots + w_{n_x} x_{n_x} b)$$

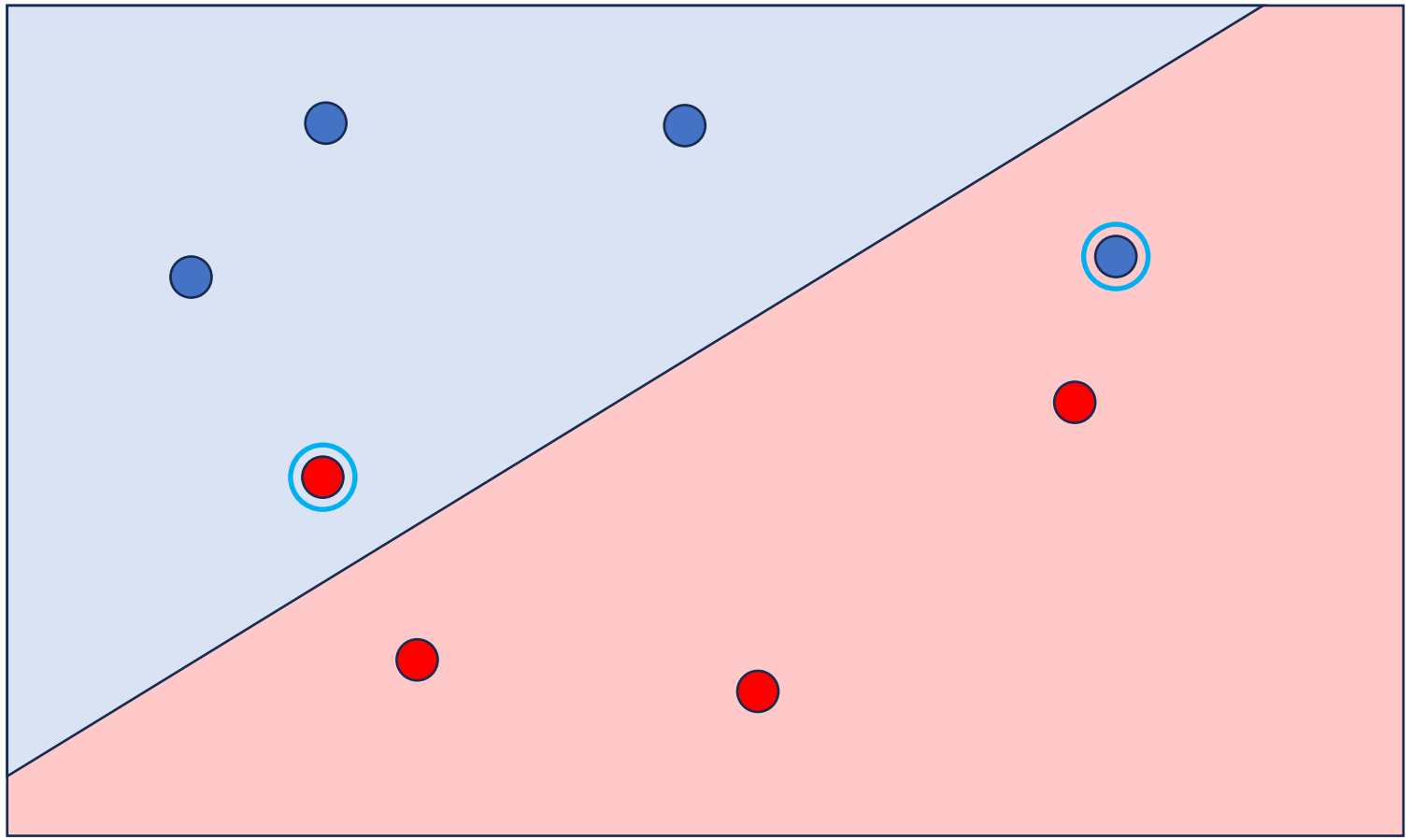
Dies entspricht im Kern der linearen Regression, allerdings bringt die logistische Funktion zusätzlich den Wert auf den Wertebereichbereich zwischen 0 und 1, wie dies für einen Wahrscheinlichkeitswert sinnvoll ist.

Kostenfunktion und Gradientenabstieg (Gradient Descent)

Beispiel: Binäre Klassifikation

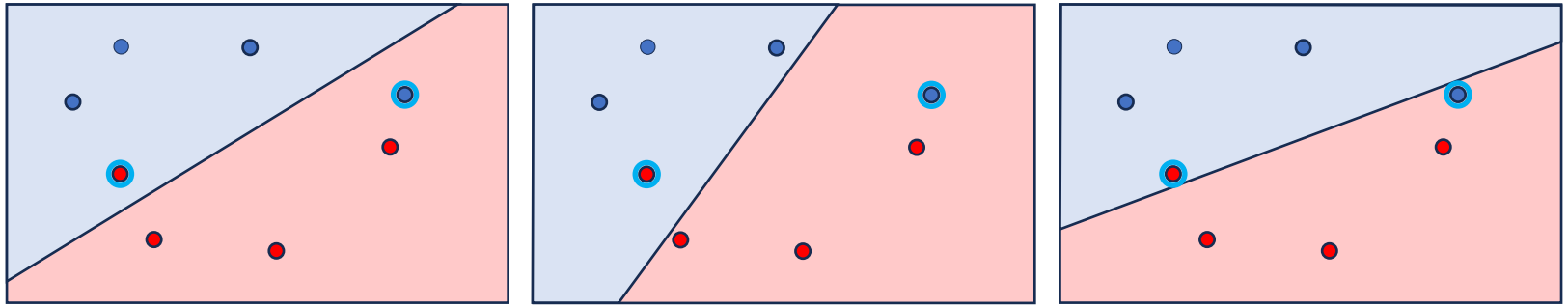
- In diesem Beispiel sind sechs Punkte richtig und zwei Punkte falsch klassifiziert.

- Wie könnte die Fehlerfunktion aussehen, welche diese Situation beschreibt?



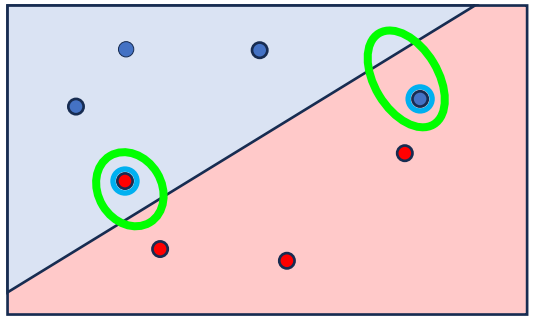
Option 1: Diskrete Fehlerfunktion

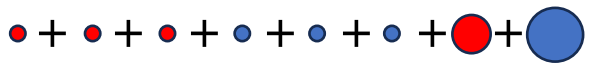
- Eine einfache Fehlerfunktion könnte einfach die Anzahl der Fehler (also der falsch klassifizierten Punkte) zählen. In unserem Beispiel würde dies einfach 2 ergeben.
- Dabei handelt es sich um eine diskrete Fehlerfunktion.
- Nachteil: Auch wenn wir die Linie verändert haben, kann die Fehlerfunktion immer noch den Wert 2 ausgeben. Wir wissen dann nicht, ob die Veränderung in die richtige Richtung geführt hat oder nicht.

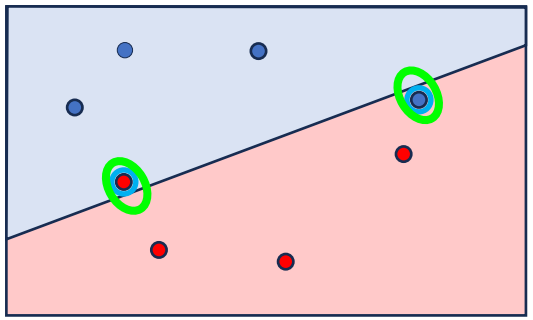


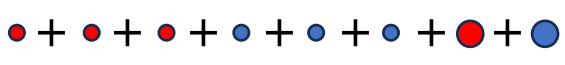
Option 2: Kontinuierliche Fehlerfunktion

- Idee: Wir konstruieren eine kontinuierliche (d.h. stetige) Fehlerfunktion. Diese ermöglicht uns festzustellen, ob eine kleine Änderung der Linie uns hilft, den Fehler zu minimieren.
- Zum Beispiel können wir die Funktion wir folgt konstruieren:
  - *Richtig klassifizierter Punkt*: kleiner Fehlerwert, der fast bei null liegt
  - *Falsch klassifizierter Punkt*: großer Fehlerwert, der sich am Abstand zwischen Punkt und der Linie orientiert
  - Der Gesamtfehler ergibt sich dann aus der Summe der Fehler der einzelnen Punkte.



Fehler = 

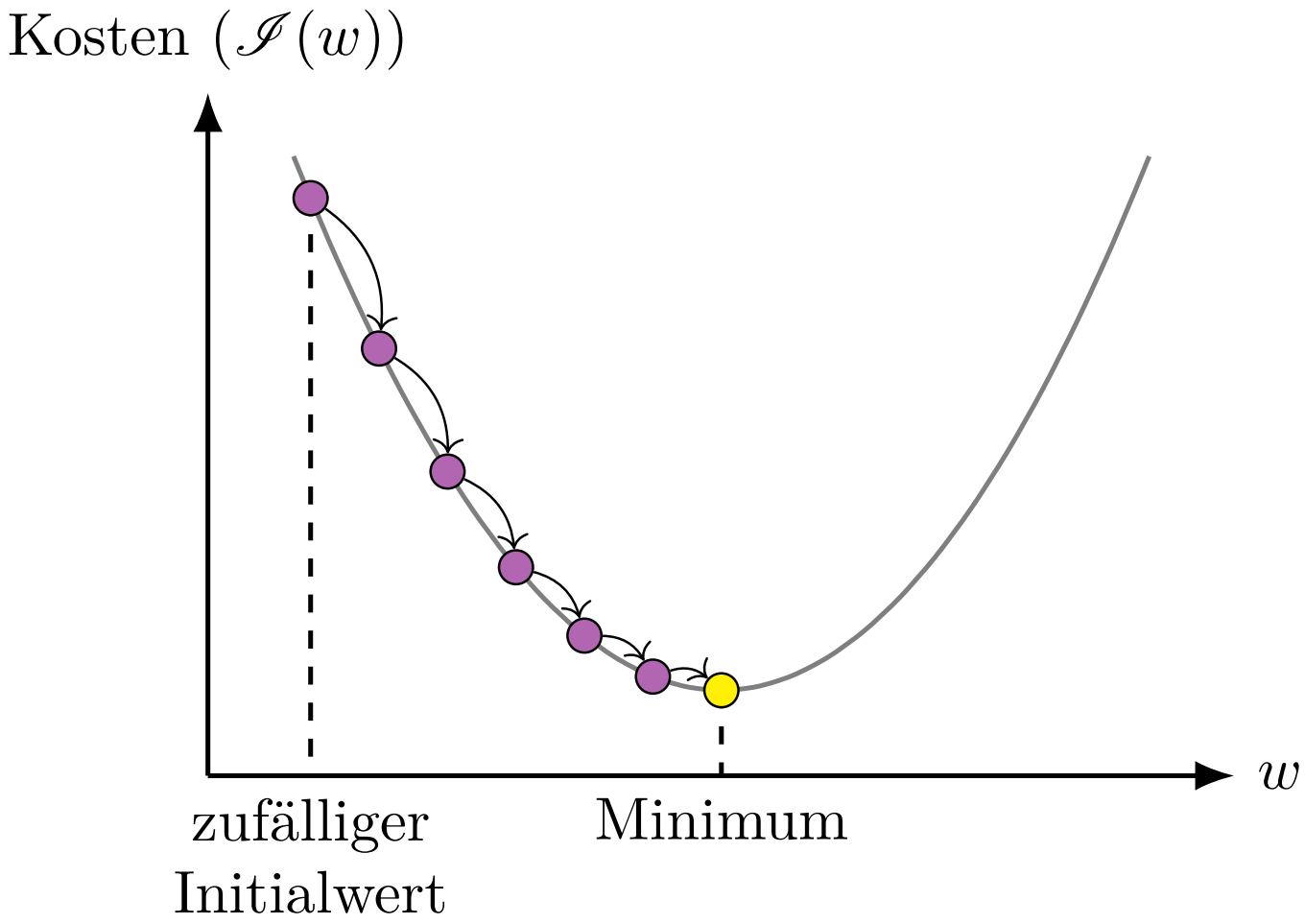


Fehler = 

Gradient Descent – Analogie

- Ein Wanderer steht auf einem Berggipfel und will möglichst schnell hinabsteigen. Es ist wolkig und das Gelände ist unübersichtlich.
- Er entscheidet sich für eine Richtung, die ihn aktuell am schnellsten nach unten bringen wird und geht diese für eine Weile.
- Danach wiederholt er den Prozess und legt die Richtung für die nächste Etappe fest, bis er unten angekommen ist.
- Die Höhe entspricht in dieser Analogie dem Fehler, den es zu minimieren gilt.

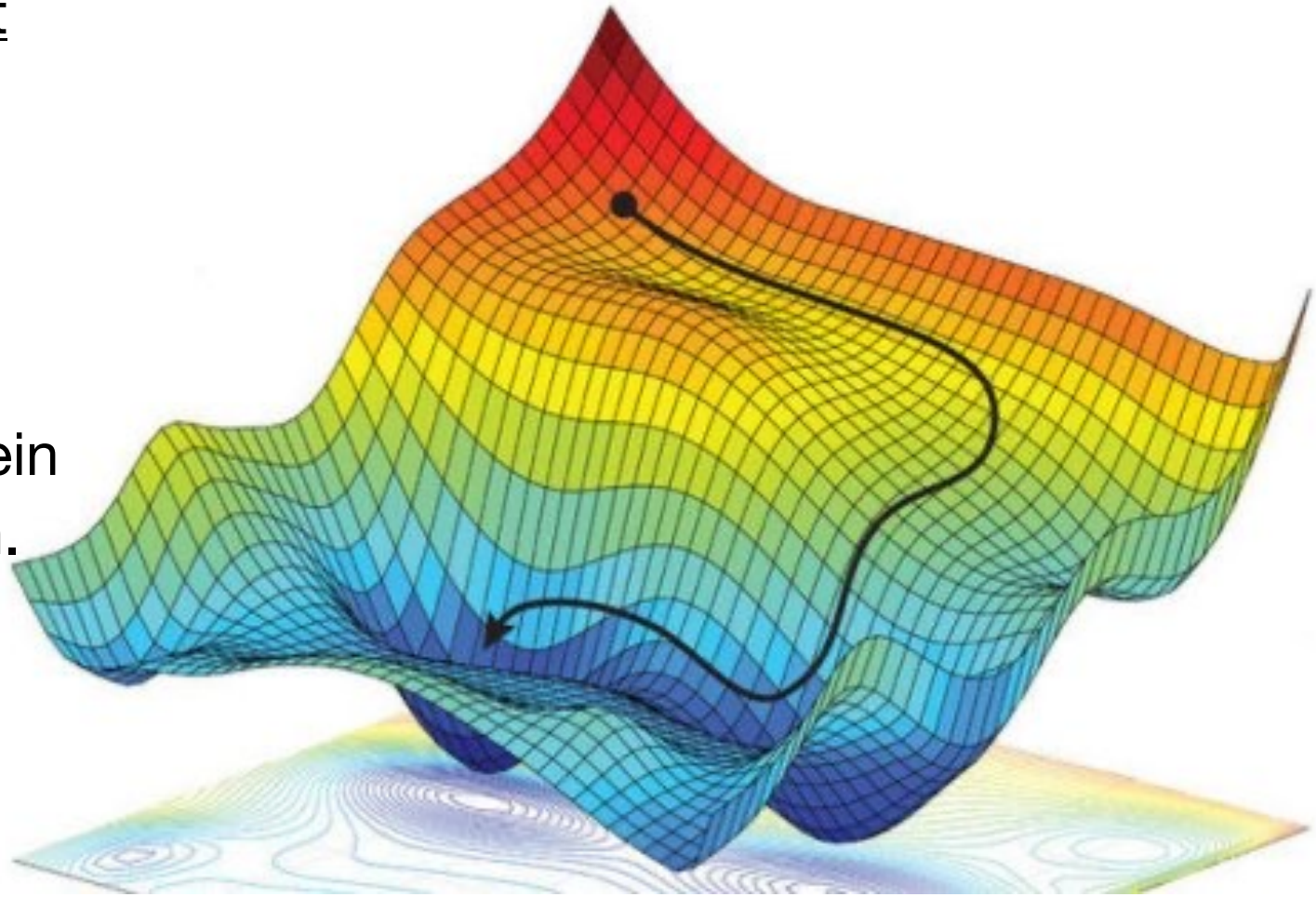
Grundprinzip von Gradient Descent



Anforderungen an die Fehlerfunktion

Um Gradientenabstieg (Gradient Decent) anzuwenden, muss die Fehlerfunktion stetig und differenzierbar sein.

Idealerweise sollte sie konvex sein und keine lokalen Minima haben.



Alexander Amini, Ava Soleimany, Sertac Karaman, and Daniela Rus. 2018. Spatial Uncertainty Sampling for End-to-End Control. *arXiv* (2018). https://doi.org/10.48550/arxiv.1805.04829

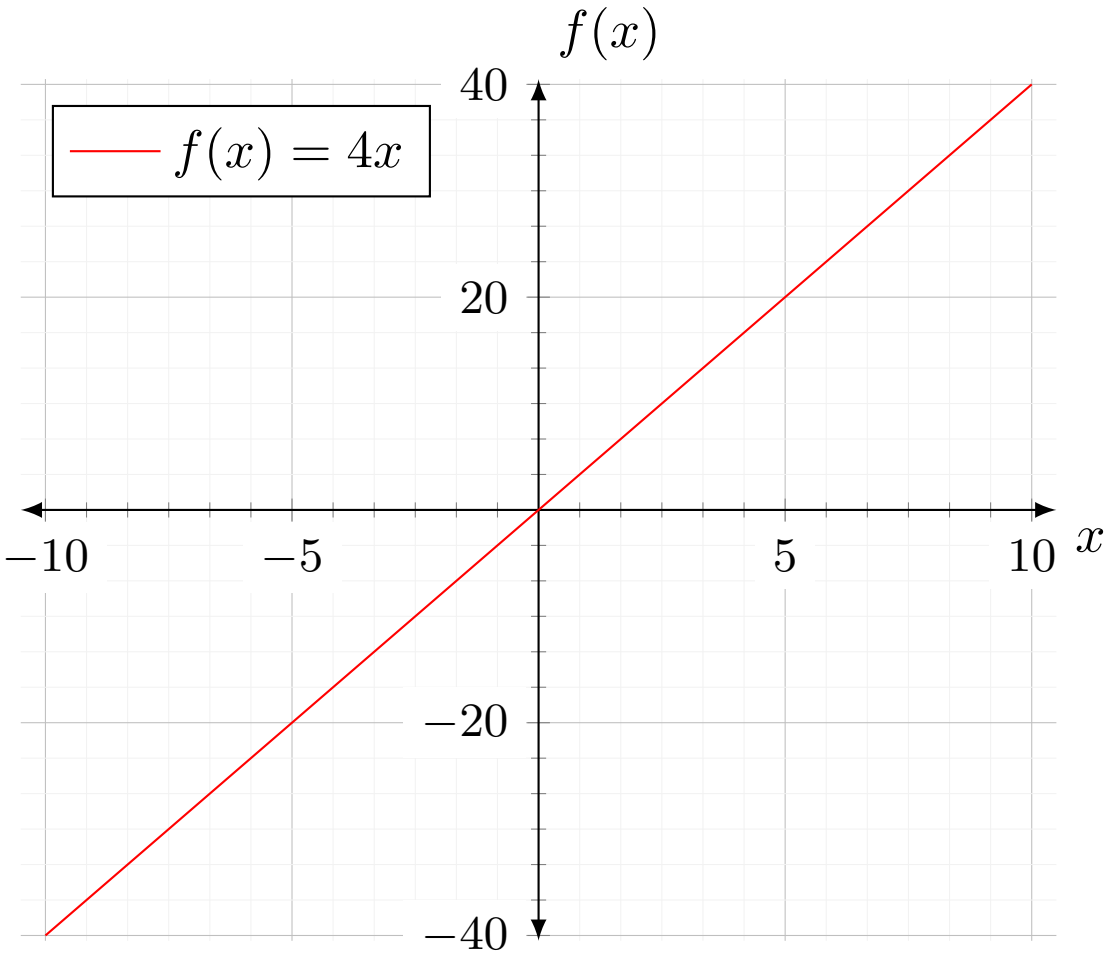
Lokale Minima

- Bei Gradient Decent kann es sein, dass man in einem lokalen Minimum „stecken bleibt“.
- Es gibt verschiedene Ansätze, um dies zu verhindern.
- In der Praxis stellen solche lokalen Minima manchmal bereits sehr gute Lösungen dar.

https://www.facebook.com/convolutionalmemes/

Wiederholung: Ableitung einer Funktion

Ableitung einer Geraden

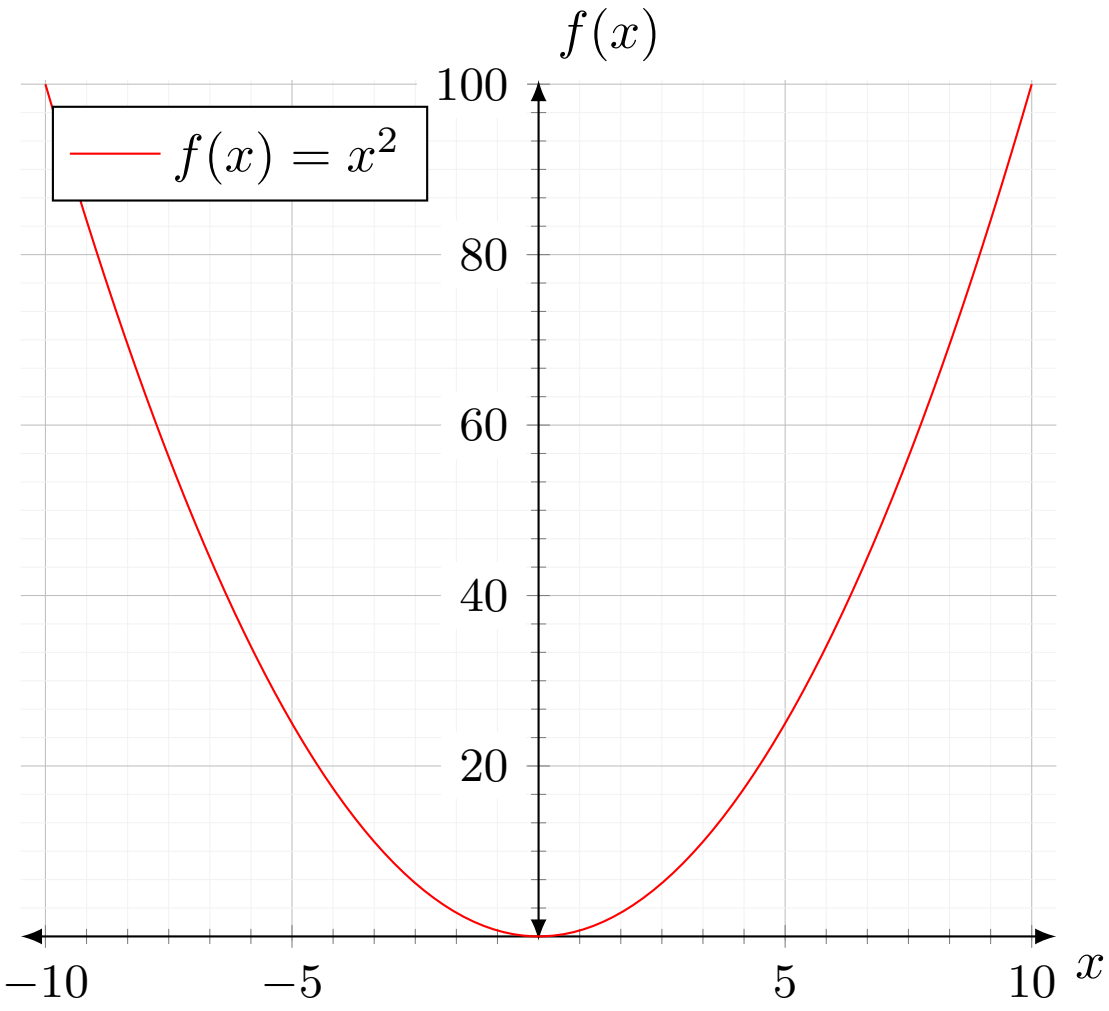


$x = 3 \Rightarrow f(x) = 12$
$x = 3.001 \Rightarrow f(x) = 12.004$

Eine kleine Änderung von x bewirkt eine 4-fache Veränderung von y.
$\Rightarrow$ Die Steigung der Geraden ist 4.

Schreibweisen:
$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx} f(x) = 4$

Verallgemeinerung



$x = 2 \Rightarrow f(x) = 4$
$x = 2.001 \Rightarrow f(x) = 4.004001$

Eine kleine Änderung von x bewirkt eine 4-fache Veränderung von y. $\Rightarrow$ Die Steigung der Kurve ist 4 (bei $x = 2$).

$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx} f(x) = 4$ (bei $x = 2$)

$x = 5 \Rightarrow f(x) = 25$
$x = 5.001 \Rightarrow f(x) = 25.010001$

Eine kleine Änderung von x bewirkt eine 10-fache Veränderung von y. $\Rightarrow$ Die Steigung der Kurve ist 10 (bei $x = 5$).

$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx} f(x) = 10$ (bei $x = 5$)

Allgemeine Regel:
$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx} f(x) = 2x$

Beispiele für Ableitungsregeln

$$f(x) = x^2$$
$$f'(x) = \frac{d}{dx} f(x) = 2x$$

$$f(x) = x^3$$
$$f'(x) = \frac{d}{dx} f(x) = 3x^2$$

$$f(x) = \log_e(x) = \ln(x)$$
$$f'(x) = \frac{d}{dx} f(x) = \frac{1}{x}$$

$$f(x) = e^x$$
$$f'(x) = \frac{d}{dx} f(x) = e^x$$

Partielle Ableitung

$$f(x_1, x_2) = 5x_1^3 + 3x_2^2$$

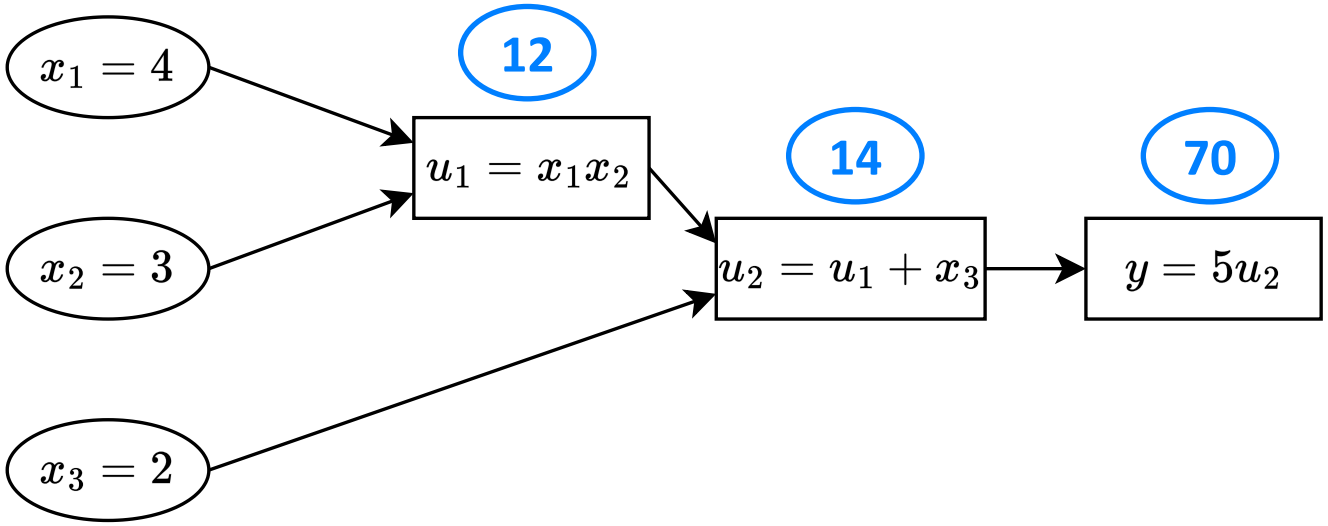
$$\frac{\partial}{\partial x_1} f(x) = 15x_1^2$$

$$\frac{\partial}{\partial x_2} f(x) = 6x_1$$

Berechnung der Ableitung im Berechnungsgraph

Berechnungsgraph

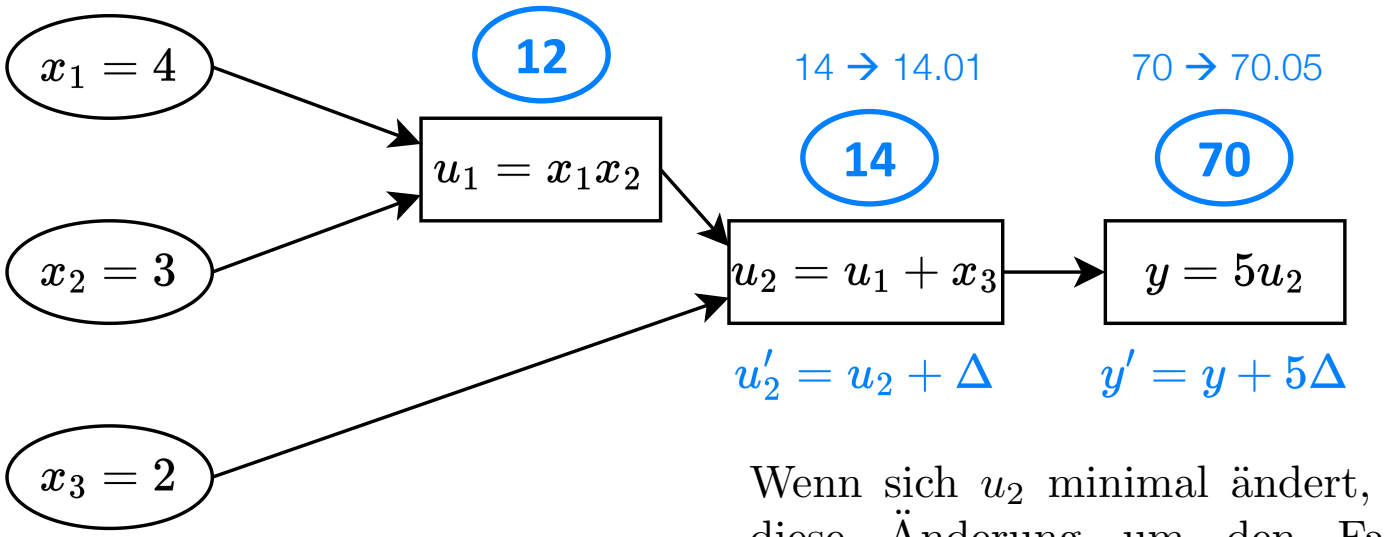
Die Funktion $f(x_1, x_2, x_3) = 5(x_1 x_2 + x_3)$ kann man schrittweise berechnen. Daraus ergibt sich ein Berechnungsgraph.



$$u_1 = x_1 x_2$$
$$u_2 = u_1 + x_3$$
$$y = 5u_2$$

Berechnung der Funktion von links nach rechts (vorwärts)

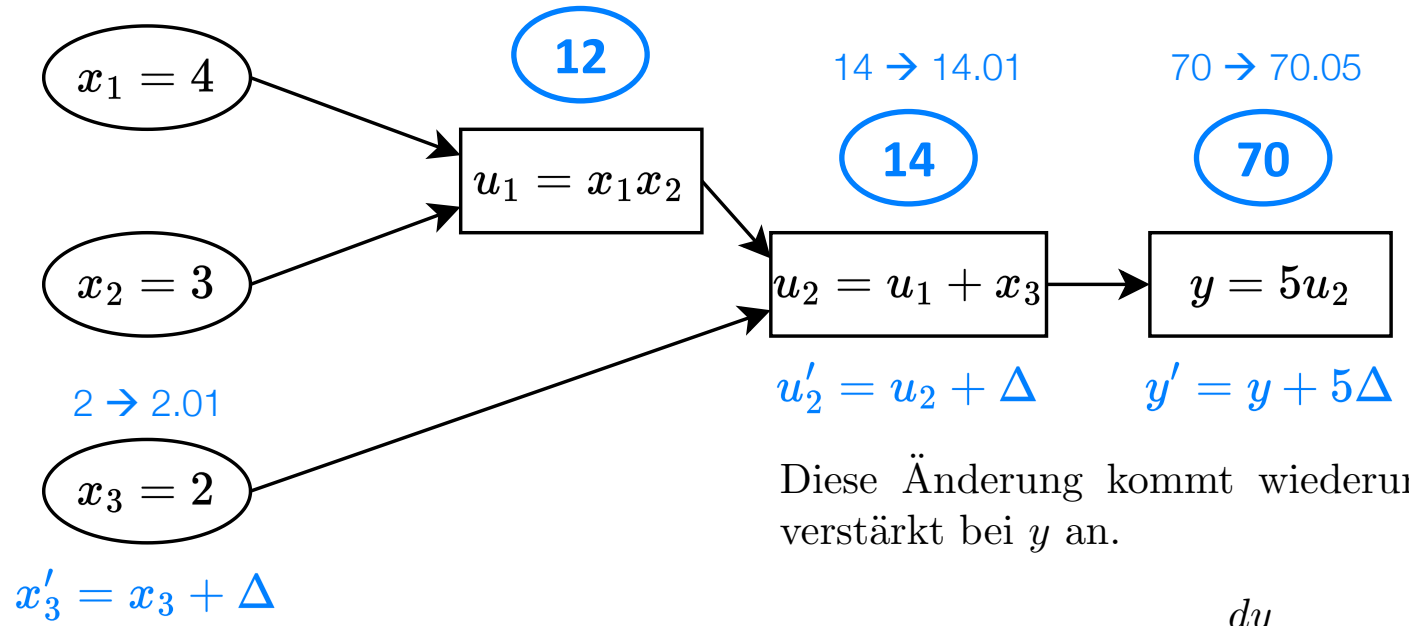
Berechnung der Ableitung: Beispiel „u_2“



Wenn sich u_2 minimal ändert, kommt diese Änderung um den Faktor 5 verstärkt bei y an.

$$\frac{dy}{du_2} = 5$$

Berechnung der Ableitung: Beispiel „x_3“



Wenn sich x_3 minimal ändert, kommt diese Änderung direkt bei u_2 an.

$$\frac{du_2}{dx_3} = 1$$

Diese Änderung kommt wiederum um den Faktor 5 verstärkt bei y an.

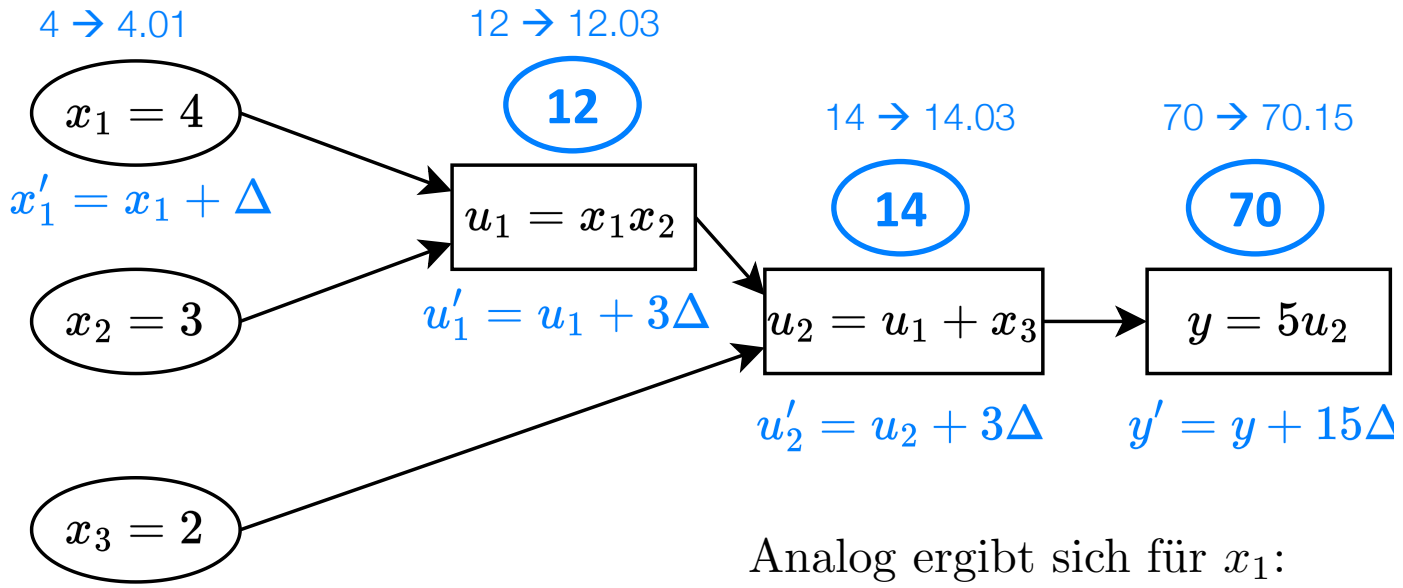
$$\frac{dy}{du_2} = 5$$

Insgesamt kommt eine minimale Änderung von x_3 also um den Faktor 5 verstärkt bei y an.

$$\frac{dy}{dx_3} = \frac{dy}{du_2}\frac{du_2}{dx_3} = 5$$

Kettenregel

Berechnung der Ableitung: Beispiel „x_1“



Analog ergibt sich für x_1:

$$\frac{dy}{dx_1} = \frac{dy}{du_2}\frac{du_2}{du_1}\frac{du_1}{dx_1} = 5 * 1 * 3 = 15$$

d.h. eine minimale Änderung bei x_3 kommt um den Faktor 15 verstärkt bei y an.

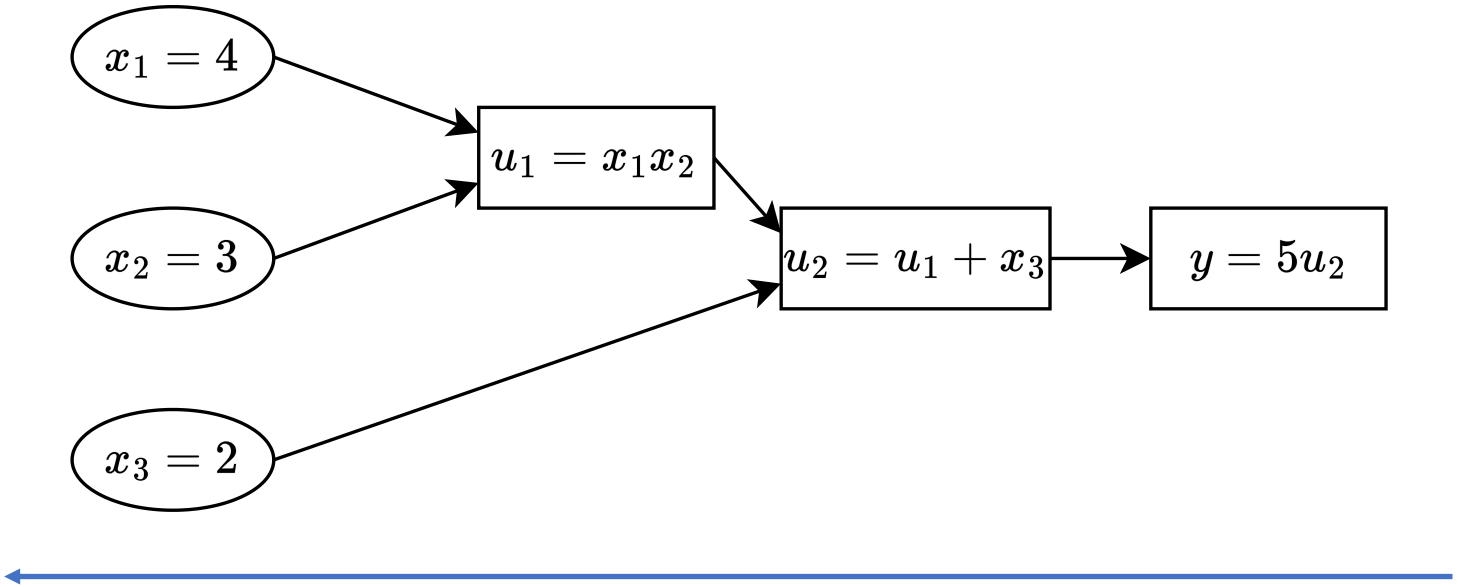
Fazit

Kurzschreibweise:
(z.B. im Source Code)

„dx1“ $\frac{dy}{dx_1} = 15$

„dx2“ $\frac{dy}{dx_2} = 20$

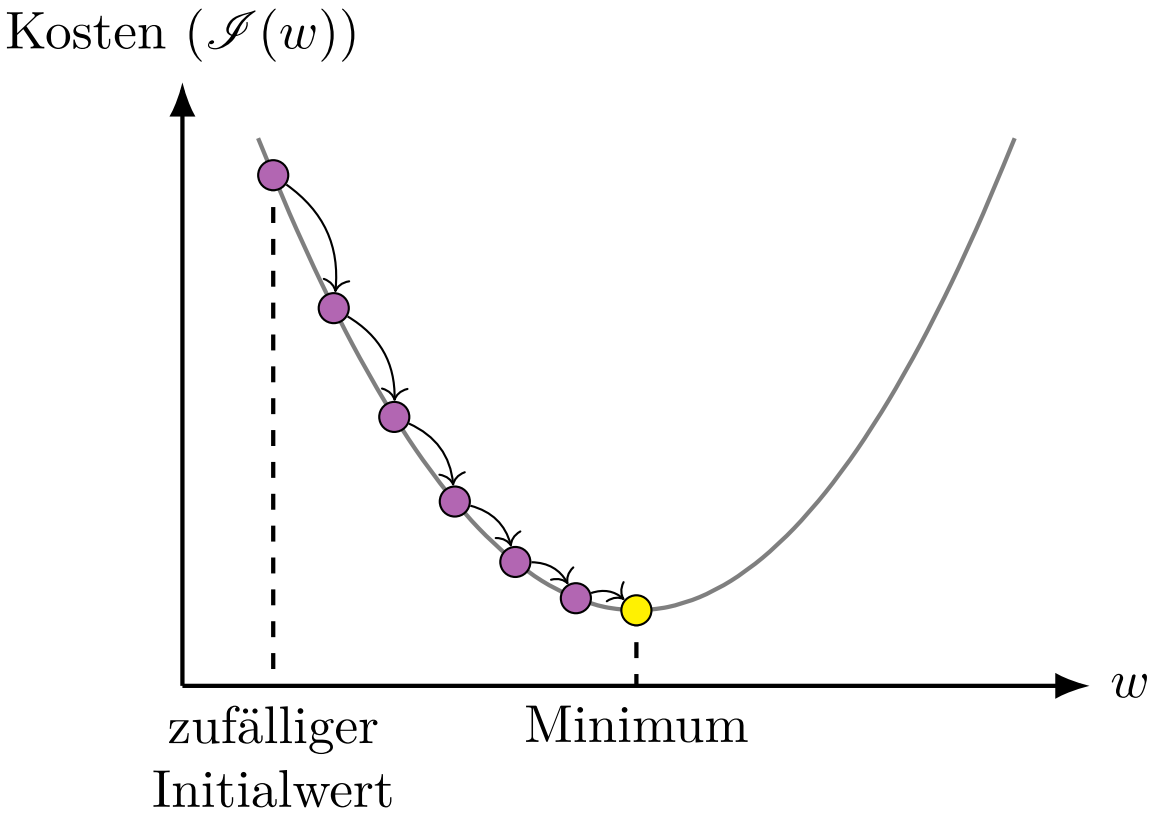
„dx3“ $\frac{dy}{dx_3} = 3$



Die Berechnung der Ableitung kann im Berechnungsgraph effizient über die Kettenregel von rechts nach links erfolgen (rückwärts).

Die Ableitung im Gradient Descent

Grundprinzip von Gradient Descent



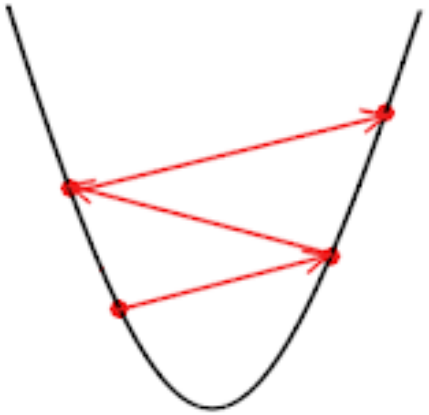
Wiederhole:

$$w := w - \alpha \frac{d\mathscr{I}(w)}{dw}$$

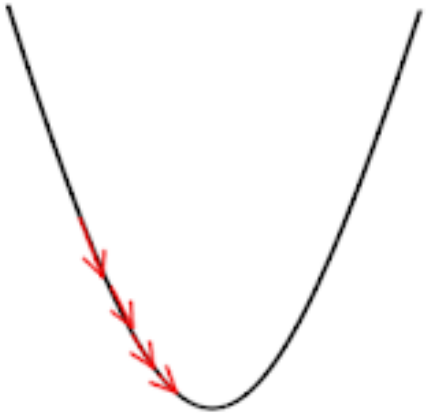
α: Lernrate

Wahl der Lernrate

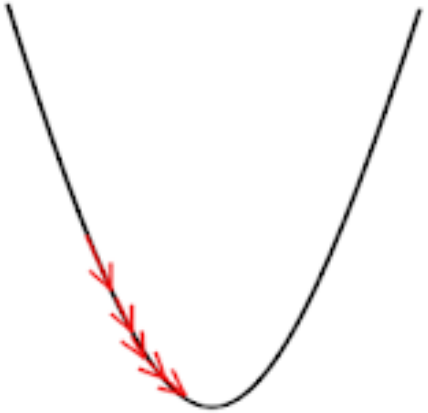
zu groß



richtig



zu klein



https://www.kaggle.com/code/residentmario/tuning-your-learning-rate

Kostenfunktion bei der logistischen Regression

Fehlerfunktion bei der logistischen Regression

Für eine Menge von m Trainingsdaten, bestehend aus je einem Paar mit Eingangsvektor x und Ausgabeklasse y: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

ergibt sich die Formel für logistische Regression für das i-te Paar:

$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b) \text{ mit } \sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$$

Gesucht ist eine Fehlerfunktion, welche die Ausgabeklasse aus den Trainingsdaten $y^{(i)}$ mit der vom Modell berechneten Ausgabeklasse $\hat{y}^{(i)}$ vergleicht.

Wir verwenden folgende Fehlerfunktion:

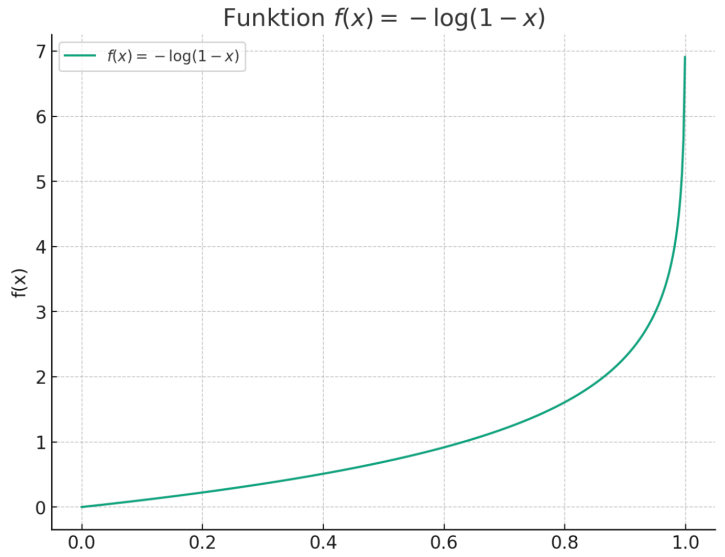
$$\mathscr{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$$

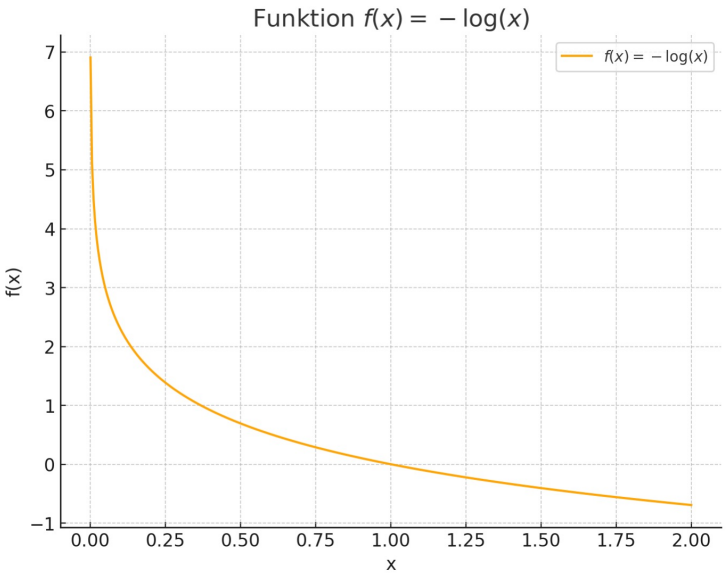
Achung! Die Fehlerfunktion vergleicht nur y und $\hat{y}$ von einem Traingsdaten-Paar, nicht über alle m Trainingsdaten hinweg.

Intuitive Beurteilung der Fehlerfunktion

Man kann sich intuitiv klar machen, warum diese Fehlerfunktion geeignet ist:

1. Fall: $y = 0$: $\mathscr{L}(\hat{y}, y) = -\log(1 - \hat{y})$ [Der andere Term fällt weg.]
   - Der Fehlerwert wird sehr groß (geht also gegen ∞), wenn $\hat{y}$ groß wird (also gegen 1 geht).
   - Der Feherwert wird sehr klein (geht also gegen 0), wenn $\hat{y}$ klein wird (also gegen 0 geht).
2. Fall: $y = 1$: $\mathscr{L}(\hat{y}, y) = -\log \hat{y}$ [Der andere Term fällt weg.]
   - Der Fehlerwert wird sehr klein (geht also gegen 0), wenn $\hat{y}$ groß wird (also gegen 1 geht).
   - Der Feherwert wird sehr groß (geht also gegen ∞), wenn $\hat{y}$ klein wird (also gegen 0 geht).





Kostenfunktion bei der logistischen Regression

Die Kostenfunktion $\mathscr{I}(w, b)$ berechnet den durchschnittlichen Fehler über alle Trainingsdaten-Paare hinweg:

$$\mathscr{I}(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathscr{L}(\hat{y}^{(i)}, y^{(i)})$$

$$= \frac{1}{m}\sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)} \right]$$

Ziel: w und b finden, so dass $\mathscr{I}(w, b)$ minimal wird.

Herleitung der Kostenfunktion (Schritt 1)

$$\hat{y} = \sigma(w^T x + b)$$

$\hat{y}$ wird interpretiert als die Wahrscheinlichkeit, dass ein Eingabevektor der Klasse angehört, d.h.

$$\hat{y} = P(y = 1|x)$$
$$1 - \hat{y} = P(y = 0|x)$$

D.h. wir benötigen eine Funktion $P(y|x)$ mit

$$P(y|x) = \hat{y} \quad (\text{falls } y = 1)$$
$$P(y|x) = 1 - \hat{y} \quad (\text{falls } y = 0)$$

Dieses Verhalten hat folgende Funktion

$$P(y|x) = \hat{y}^y (1 - \hat{y})^{(1-y)}$$

wie die beiden folgenden Fälle zeigen:

$$P(y|x) = \hat{y}^0 (1 - \hat{y})^{1-0} = 1 \cdot (1 - y) = 1 - \hat{y} \quad (\text{falls } y = 0)$$
$$P(y|x) = \hat{y}^1 (1 - \hat{y})^{1-1} = \hat{y} \cdot (1 - \hat{y})^0 = \hat{y} \quad (\text{falls } y = 1)$$

Herleitung der Kostenfunktion (Schritt 2)

Jetzt wendet man den Logarithmus an:

$$\begin{aligned}\log\left(P(y|x)\right) &= \log\left(\hat{y}^y \cdot (1-\hat{y})^{(1-y)}\right) \\ &= y \log \hat{y} + (1-y)\log(1-\hat{y}) \\ &= -\mathscr{L}(\hat{y}, y)\end{aligned}$$

Dies ist möglich, da die Logarithmusfunktion streng monoton steigend ist. Anschaulich: Kleine Werte werden auf kleine Werte abgebildet, große Werte auf große Werte.

Die Maximierung der Wahrscheinlichkeit entspricht der Minimierung der Kostenfunktion.

Herleitung der Kostenfunktion (Schritt 3)

Die Log-Likelihood-Funktion wird maximiert, um die Modellparameter zu finden, welche die Daten aus dem Datensatz am besten erklären. Hierdurch wird das Modell "trainiert". (Unter der Annahme, dass die Daten unabhängig und identisch verteilt sind.)

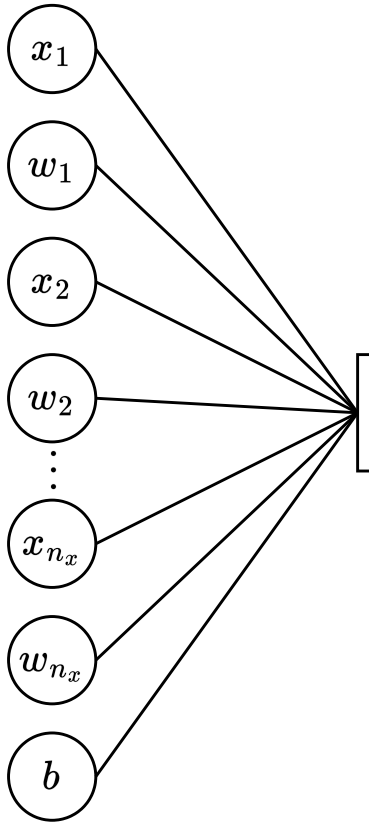
$$\log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)})$$
$$= -\sum_{i=1}^{m} \mathscr{L}(\hat{y}^{(i)}, y^{(i)})$$

Die Maximierung der Log-Likelihood-Funktion entspricht der Minimierung der Kostenfunktion.

$$\mathscr{J}(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}(\hat{y}^{(i)}, y^{(i)})$$

Gradient Descent bei der Logistischen Regression

Herleitung Berechnungsgraph



$$\mathscr{L}(a, y) = -(y \log a + (1 - y) \log(1 - a)) \quad (1)$$

$$a = \sigma(z) \quad (2)$$

$$z = w^T x + b \quad (3)$$

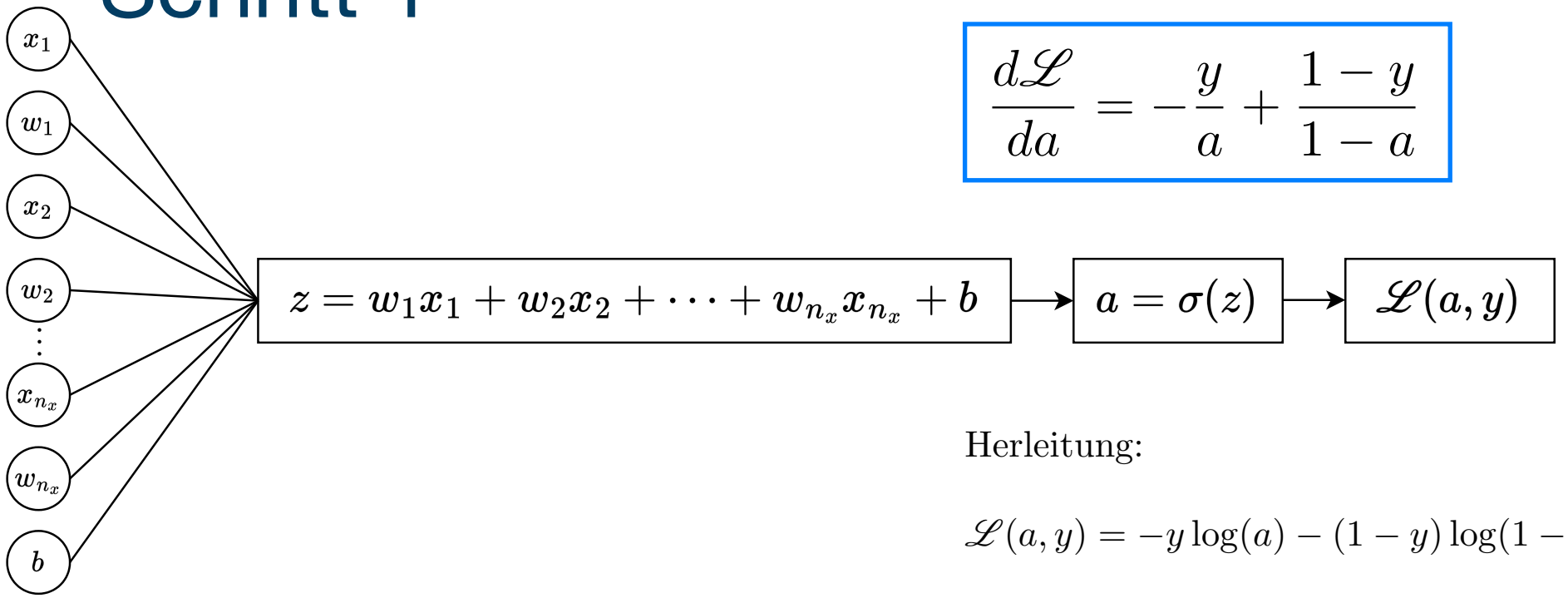
$$z = w_1 x_1 + w_2 x_2 + \cdots + w_{n_x} x_{n_x} + b \rightarrow a = \sigma(z) \rightarrow \mathscr{L}(a, y)$$

Ziel: Wir leiten eine einfache Formel für folgende Werte her, um sie im Gradient Descent verwenden zu können, denn wir wollen w_i und b zielgerichtet verändern, um die Kostenfunktion $\mathscr{L}$ zu minimieren:

$dw_i = \frac{d\mathscr{L}}{dw_i}$ (für alle i)
$db = \frac{d\mathscr{L}}{db}$

Hierzu nutzen wir die Kettenregel im Berechnungsgraph.

Schritt 1



$$\frac{d\mathscr{L}}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

Herleitung:

$$\mathscr{L}(a, y) = -y\log(a) - (1-y)\log(1-a)$$

$$\frac{\partial}{\partial a}(-y\log(a)) = -\frac{y}{a}$$

$$\frac{\partial}{\partial a}(-(1-y)\log(1-a)) = (1-y)\frac{1}{1-a}\cdot(-1) = \frac{1-y}{1-a}$$

Zusammengefasst haben wir:

$$\frac{d\mathscr{L}}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

Ableitung der logistischen Funktion

Ziel: Ableitung $\frac{d\sigma(x)}{dx}$

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

$$= \frac{1}{1+\frac{1}{e^x}} \tag{2}$$

$$= \frac{e^x}{e^x+1} \tag{3}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^x\left(e^x+1\right)-e^x e^x}{\left(e^x+1\right)^2} \tag{4}$$

$$= \frac{e^{2x}+e^x-e^{2x}}{\left(e^x+1\right)^2} \tag{5}$$

$$= \frac{e^x}{\left(e^x+1\right)^2} \tag{6}$$

$$= \frac{e^{2x}+e^x-e^{2x}}{(e^x+1)^2} \tag{7}$$

$$= \frac{e^x(e^x+1)-e^{2x}}{(e^x+1)^2} \tag{8}$$

$$= \frac{e^x}{e^x+1} - \frac{e^{2x}}{(e^x+1)^2} \tag{9}$$

$$= \frac{e^x}{e^x+1}\left(1-\frac{e^x}{e^x+1}\right) \tag{10}$$

$$= \sigma(x)(1-\sigma(x)) \tag{11}$$

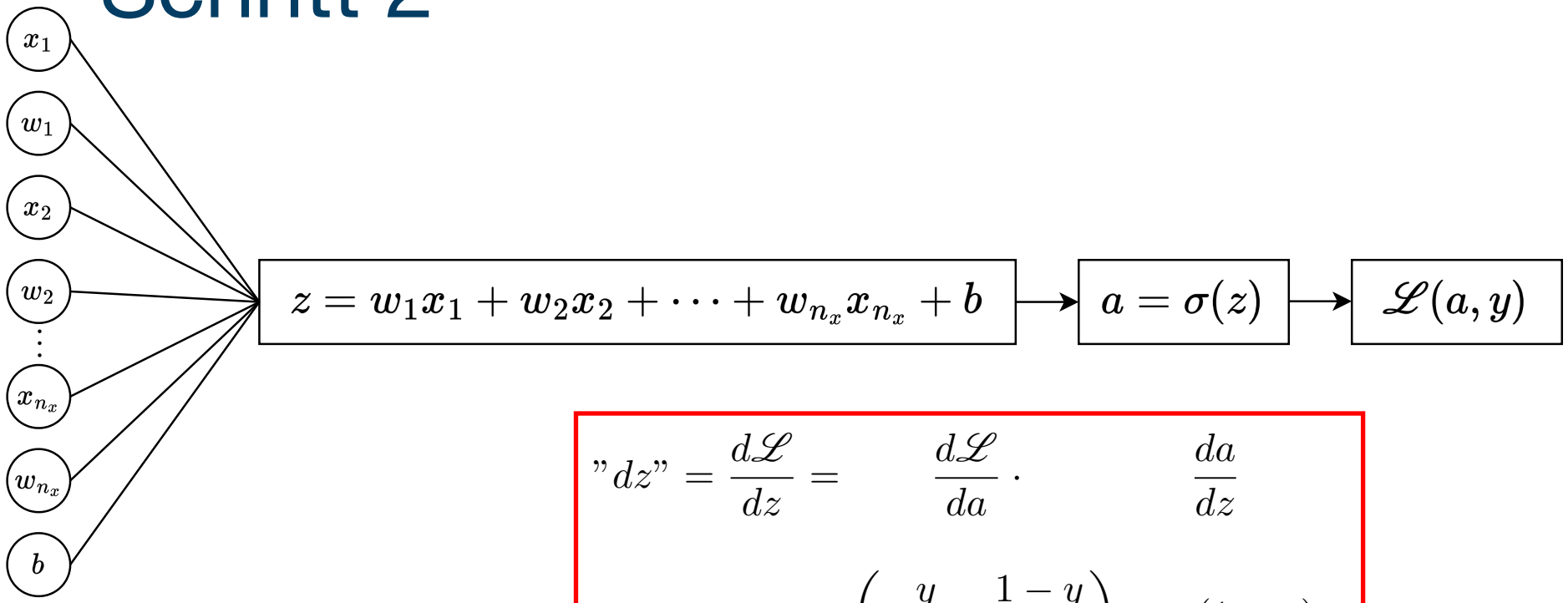
Quotientenregel:

$$\left(\frac{u(x)}{v(x)}\right)' = \frac{u'(x)v(x)-u(x)v'(x)}{(v(x))^2}$$

oder:

$$\frac{d}{dx}\left(\frac{u(x)}{v(x)}\right) = \frac{\frac{d}{dx}u(x)\cdot v(x)-u(x)\cdot\frac{d}{dx}v(x)}{(v(x))^2}$$

Schritt 2



$$\text{"}dz\text{"} = \frac{d\mathscr{L}}{dz} = \frac{d\mathscr{L}}{da} \cdot \frac{da}{dz}$$

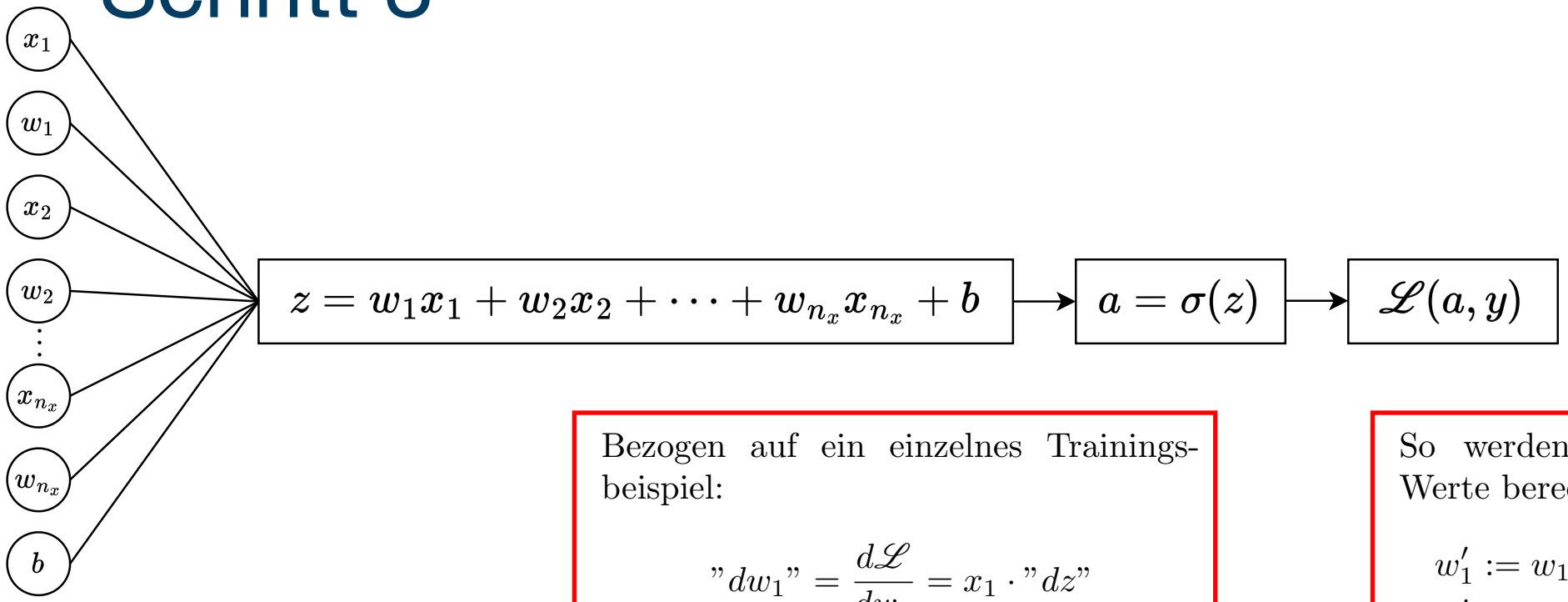
$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) \cdot a \cdot (1-a)$$

$$= -\frac{y}{a}a(1-a) + \frac{1-y}{1-a}a(1-a)$$

$$= -y + ay + a - ay$$

$$= a - y$$

Schritt 3



Bezogen auf ein einzelnes Trainingsbeispiel:

$$"dw_1" = \frac{d\mathscr{L}}{dw_1} = x_1 \cdot "dz"$$

$$"dw_2" = \frac{d\mathscr{L}}{dw_2} = x_2 \cdot "dz"$$

$$\vdots$$

$$"dw_n" = \frac{d\mathscr{L}}{dw_n} = x_n \cdot "dz"$$

$$"db" = "dz"$$

So werden die neuen Werte berechnet:

$$\begin{aligned} w_1' &:= w_1 - \alpha \cdot dw_1 \\ w_2' &:= w_2 - \alpha \cdot dw_2 \\ &\vdots \\ w_n' &:= w_n - \alpha \cdot dw_n \\ b' &:= b - \alpha \cdot db \end{aligned}$$

Mit *m* Trainingsbeispielen

Die Kostenfunktion berechnet sich als Durchschnitt der Fehlerfunktion über alle Beispiele:

$$\mathscr{I}(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathscr{L}(\hat{y}^{(i)}, y^{(i)})$$

$$\hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$$\frac{\partial}{\partial w_i}\mathscr{I}(w,b) = \frac{1}{m}\sum_{j=1}^{m}\frac{\partial}{\partial w_i}\mathscr{L}(\hat{y}^{(j)}, y^{(j)})$$

$$= \frac{1}{m}\sum_{j=1}^{m} dw_i^{(j)}$$

$$\frac{\partial}{\partial b}\mathscr{I}(w,b) = \frac{1}{m}\sum_{j=1}^{m}\frac{\partial}{\partial b}\mathscr{L}(\hat{y}^{(j)}, y^{(j)})$$

$$= \frac{1}{m}\sum_{j=1}^{m} db^{(j)}$$

Umsetzung in Pseudo-Code

```
Für jede Iteration:
    // Initialisierung
    I = 0
    dw_1 = 0
    dw_2 = 0
    db = 0

    // durch alle Trainingsbispiele gehen
    Für i = 1 bis m:
        // Prediction a für i-tes Trainingsbeispiel berechnen
        z = vektormult(w, x[i]) + b
        a = sigmoid(z)

        // Fehler berechnen
        I = I - (y[i] * log(a) + (1 - y[i])*log(1 - a))

        // dz berechnen
        dz = a - y[i]

        // partielle Ableitungen berechnen und aufsummieren
        dw_1 = dw_1 + x[i][1] * dz
        dw_2 = dw_2 + x[i][2] * dz
        db = db + dz

// Durchschnitt über alle Trainingsbeispiele berechnen
I = I / m
dw_1 = dw_1 / m
dw_2 = dw_2 / m
db = db / m

// Parameter aktualisieren
w_1 = w_1 - alpha * dw_1
w_2 = w_2 - alpha * dw_2
b = b - alpha * db
```

Hier nur mit den Parametern w_1, w_2 und b.