Alles auf eine Skala (Normalisierung)

Wohnfläche 350, Zimmerzahl 4 — warum so ungleiche Zahlen ein Training kippen lassen, und wie Min-Max und Z-Score das reparieren. Mit Live-Beweis: zwei Trainings im Direktvergleich.

Kapitel online

Praxis · Kapitel 17 von 50 · ca. 15 Min. · 150 XP

Einfach gesagt: Wohnfläche und Zimmerzahl liegen in ganz unterschiedlichen Größenordnungen — bringst du sie nicht auf dieselbe Skala, kippt das Training.

Sie kommt garantiert: *„Müssen die Zahlen im Tensor nicht zwischen 0 und 1 liegen?“* Kurze Antwort aus Kapitel 010: Nein, ein Tensor darf alles enthalten. Lange Antwort: **Fürs Training** ist Skalieren trotzdem oft der Unterschied zwischen „läuft sauber“ und „explodiert“. Heute siehst du beides live — nebeneinander.

Was du hier lernst

- **Warum ungleiche Spalten ein Training kippen** — 320 gegen 7, und der Gradient läuft aus dem Ruder
- **Min-Max** — jede Spalte auf 0–1 bringen, als Tensor-Rechnung zum Mitrechnen
- **Z-Score** — Mittelwert 0, Streuung 1: die robustere Variante, wenn Ausreißer drin sind
- **Der Live-Beweis** — zweimal dasselbe Training, einmal roh, einmal skaliert
- **Die Profi-Regel** — Skalierungs-Werte nur aus den Trainingsdaten, danach unverändert weiterbenutzen

Für den Unterricht

Einstieg (3 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (9 Min): die interaktiven Module auf herrki.de/135_daten_vorbereiten ausprobieren.

Sichern mit Quiz (3 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Das Problem: Zahlen in völlig verschiedenen Ligen

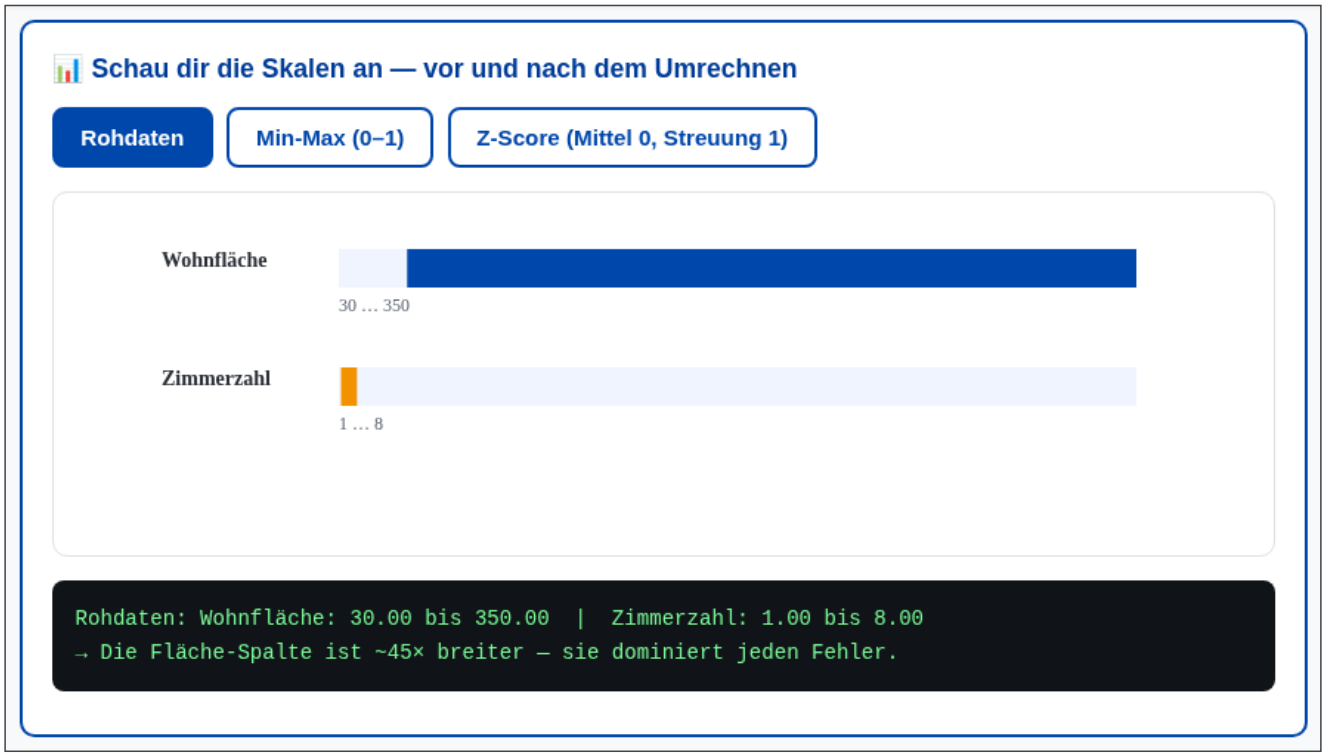
Zurück zu den Wohnungspreisen aus „Multiple Regression“. Unsere Merkmale spielen in verschiedenen Größenordnungen:

Merkmal	kleinster Wert	größter Wert	Spannweite
Wohnfläche [m²]	30	350	**320**
Zimmerzahl	1	8	**7**

Für das Modell bedeutet das: Ein kleiner Fehler bei der Wohnfläche erzeugt riesige Zahlen im Loss, ein Fehler bei der Zimmerzahl fast nichts. **Die große Spalte dominiert alles** — und der Gradient (Kapitel 118) wird in eine Richtung gigantisch, in die andere winzig. Mit einer Lernrate, die für beide passt? Gibt es dann nicht.

Am Gerät ausprobieren: herrki.de/135_daten_vorbereiten#abschnitt-2

Direkt zum Modul

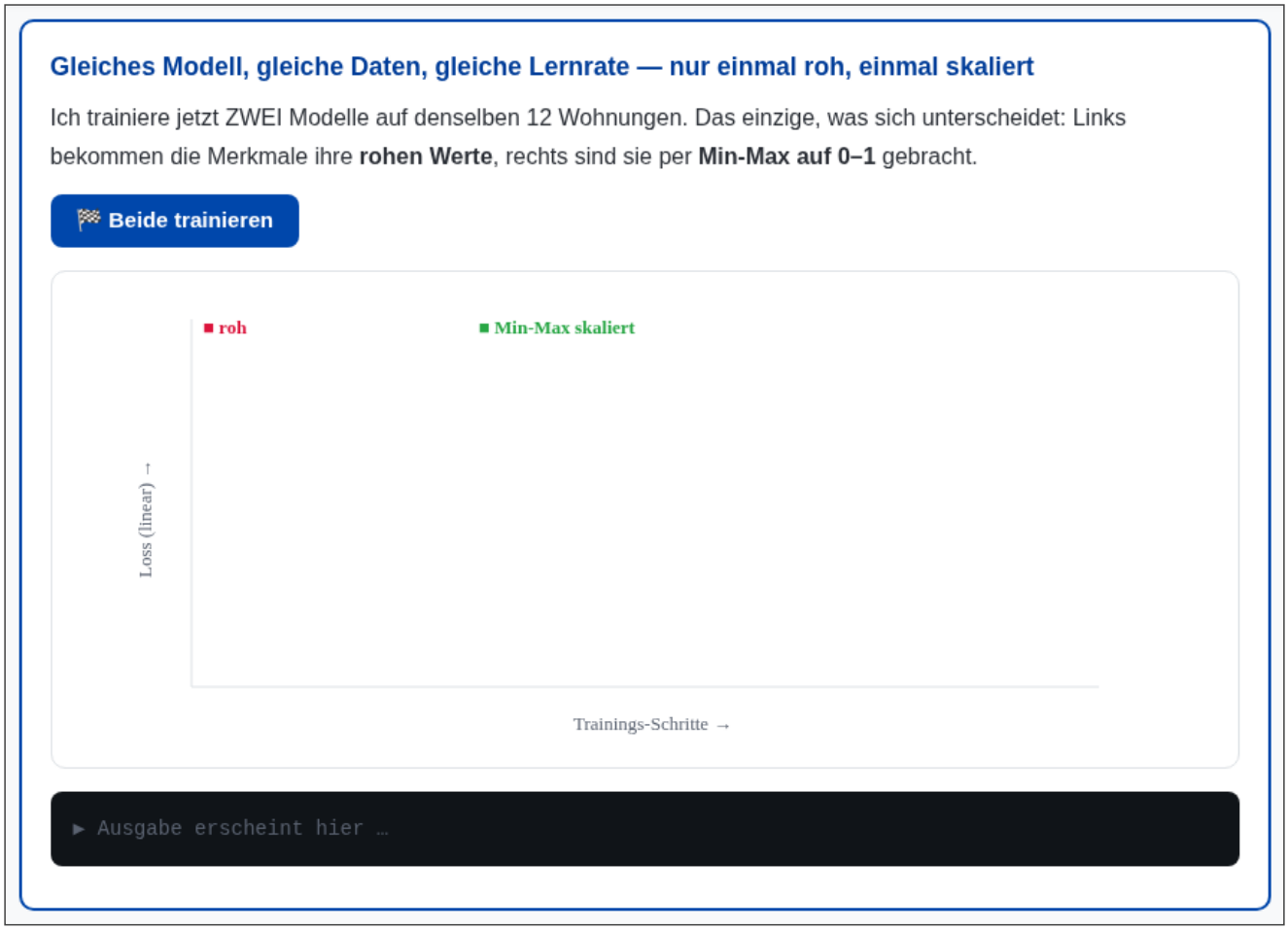


Interaktives Modul – am Gerät ausprobieren

Der Live-Beweis: zwei identische Trainings

Am Gerät ausprobieren: herrki.de/135_daten_vorbereiten#abschnitt-3

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Das war kein Trick: Das „rohe“ Training bekommt bei jedem Schritt einen riesigen Gradienten aus der 320er-Spalte — die Schritte springen über das Tal, schaukeln sich auf (du kennst das aus der Schüssel in Kapitel 118), und der Loss läuft gegen unendlich bzw. `NaN`. In der Praxis ist **fehlende Skalierung die häufigste Ursache für explodierende Loss-Werte**.

Runter zum Tensor

Die Umrechnung ist reine Tensor-Arithmetik (Kapitel 060) — und sie passiert **spaltenweise**. Hier 4 der 12 Wohnungen zum Mitrechnen:

```
merkmale [12,2] Fläche Zimmer
Wohnung 1 : 30 1 // die kleinste
Wohnung 6 : 110 4
Wohnung 8 : 170 5
Wohnung 12 : 350 8 // die größte

min(0) [2] : 30 1 // kleinster Wert JE SPALTE
max(0) [2] : 350 8 // größter Wert JE SPALTE
max−min [2] : 320 7 // die Spannweite aus der Tabelle oben
```

Jetzt jede Zahl durch *(x − min) / (max − min)* schicken — rechts steht die Rechnung dazu:

```
skaliert [12,2] Fläche Zimmer
Wohnung 1 : 0,00 0,00 // ( 30−30)/320 = 0 | (1−1)/7 = 0
Wohnung 6 : 0,25 0,43 // (110−30)/320 = 0,25 | (4−1)/7 = 3/7
Wohnung 8 : 0,44 0,57 // (170−30)/320 = 0,44 | (5−1)/7 = 4/7
Wohnung 12 : 1,00 1,00 // (350−30)/320 = 1 | (8−1)/7 = 1
```

Aus 320 gegen 7 sind zwei Spalten von 0 bis 1 geworden — dieselbe Wohnung, dieselbe Rangfolge, nur ohne das Größen-Gefälle. Beachte: `min` und `max` sind nur [2] lang (eine Zahl pro Spalte) und werden trotzdem auf alle 12 Zeilen [12,2] angewendet — das Broadcasting aus Kapitel 060:

```
// Min-Max: jede Spalte auf 0–1 (x − min) / (max − min)
const min = merkmale.min(0), max = merkmale.max(0);
const skaliert = merkmale.sub(min).div( max.sub(min) );
```

Die zweite Methode: Z-Score

Der **Z-Score** nimmt dieselben 12 Wohnungen, nur andere Kennzahlen: statt min und max den *Mittelwert* und die *Standardabweichung* (Streuung — wie weit die Werte im Schnitt um den Mittelwert herumliegen). Die Formel lautet `(x − Mittelwert) / Streuung`, und danach hat jede Spalte Mittelwert 0 und Streuung 1. Die Zahlen landen also nicht zwischen 0 und 1, sondern meist zwischen −2 und +2 — mit Vorzeichen: minus heißt „unter dem Durchschnitt“:

```
mean(0) [2] : 153,3 4,17 // Mittelwert JE SPALTE
std(0) [2] : 101,2 2,03 // Streuung JE SPALTE

Wohnung 1 : −1,22 −1,56 // ( 30−153,3)/101,2 | (1−4,17)/2,03
Wohnung 6 : −0,43 −0,08 // (110−153,3)/101,2 | (4−4,17)/2,03
Wohnung 12 : +1,94 +1,88 // (350−153,3)/101,2 | (8−4,17)/2,03
```

Wohnung 6 liegt damit „ein gutes Stück unter dem Durchschnitt“ (−0,43) und hat fast exakt durchschnittlich viele Zimmer (−0,08). In TensorFlow.js ist auch das ein Zweizeiler — `tf.moments` liefert Mittelwert und Varianz spaltenweise in einem Rutsch:

```
// Z-Score: (x − Mittelwert) / Streuung, wieder spaltenweise (Achse 0)
const {mean, variance} = tf.moments(merkmale, 0);
const zwerte = merkmale.sub(mean).div( variance.sqrt() );
```

Welche von beiden? Min-Max presst alles hart in 0–1 — ordentlich, aber ein einziger Ausreißer (die 900-m²-Villa) drückt alle anderen Wohnungen zusammen in die Nähe von 0. **Z-Score** hat keine feste Obergrenze und steckt so etwas besser weg. Faustregel: Pixel und saubere Wertebereiche → Min-Max, echte Messdaten mit Ausreißern → Z-Score. Wirklich falsch machst du mit keiner von beiden etwas — Hauptsache, du skalierst überhaupt.

Die Profi-Regel, an der selbst Fortgeschrittene scheitern: `min` und `max` (bzw. Mittelwert/Streuung) berechnest du **NUR aus den Trainingsdaten** — und wendest sie dann unverändert auf neue Daten an. Warum das so wichtig ist, verstehst du vollständig in „Trainieren, prüfen, bestehen“.

Praktisch heißt das: **Die Skalierungs-Werte gehören zum Modell wie die Gewichte.** Du hebst sie nach dem Training auf und schickst jede neue Wohnung damit durch dieselbe Umrechnung, bevor das Modell sie zu sehen bekommt:

```
// beim Training EINMAL berechnen – und aufheben
const min = trainX.min(0), max = trainX.max(0);
const trainSkaliert = trainX.sub(min).div( max.sub(min) );

// spaeter, fuer eine neue Wohnung (95 m², 3 Zimmer): DIESELBEN min/max!
const neu = tf.tensor2d([[95, 3]]);
const preis = modell.predict( neu.sub(min).div( max.sub(min) ) );
// (95−30)/320 = 0,20 | (3−1)/7 = 0,29 → so kennt das Modell die Zahlen
```

Zurück zur Anwendung

Du kennst das Prinzip längst: In „Bilder als Tensoren“ haben wir Pixel durch 255 geteilt — Min-Max für Bilder. In „Excel/CSV → Tensoren“ stand der Vorgeschmack dazu. Jetzt weißt du, **warum**: nicht weil Tensoren es verlangen — sondern damit Gradient Descent in allen Richtungen gleich gute Schritte machen kann.

Zum Weiterlesen

- **Die Werkzeugkiste:** scikit-learn, *Preprocessing data* — scikit-learn.org/stable/modules/preprocessing.html — MinMaxScaler und StandardScaler als fertige Bausteine, samt der Trennung „erst fit auf Train, dann transform auf alles“.
- **Kurz und konkret:** Google Machine Learning Crash Course, *Normalization* — developers.google.com/machine-learning/crash-course (developers.google.com/machine-learning/crash-course/numerical-data/normalization) — welche Methode wann passt, inklusive Log-Skalierung und Clipping für schiefe Verteilungen.
- **Ausblick:** Sergey Ioffe & Christian Szegedy, *„Batch Normalization“* (2015) — arxiv.org/abs/1502.03167 — dasselbe Prinzip nicht nur vor dem Netz, sondern mitten drin, zwischen den Schichten.
- **Auf HerrKI:** 118 — Gradient Descent zum Anfassen (die Schüssel, die hier rund wird) und 235 — Trainieren, prüfen, bestehen (warum die Kennzahlen nur aus den Trainingsdaten kommen dürfen)

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Müssen Tensor-Werte grundsätzlich zwischen 0 und 1 liegen?

- Ja, sonst kann der Computer sie nicht speichern.
- ✔ **Nein — Tensoren dürfen beliebige Zahlen enthalten; skaliert wird nur fürs Training.**
- Ja, das ist Teil der Tensor-Definition.

Der Tensor ist nur der Container. Die 0–1-Umrechnung ist ein Vorbereitungs-Schritt, damit Gradient Descent stabil läuft — kein Muss, aber fast immer eine sehr gute Idee.

Frage 2: Warum kippt das Training mit unskalierten Merkmalen (30–350 vs. 1–8)?

- ✔ **Die große Spalte dominiert Loss und Gradient — für beide Richtungen gleichzeitig passt keine Lernrate.**
- Weil TensorFlow nur kleine Zahlen unterstützt.
- Weil zu viele Daten im Spiel sind.

Die Loss-Landschaft wird zu einem extrem schmalen, steilen Graben: In der einen Richtung explodieren die Schritte, in der anderen kriechen sie. Skalieren macht die Schüssel wieder rund.

Frage 3: Woraus berechnet man die Skalierungs-Werte (min/max bzw. Mittelwert)?

- Aus allen Daten, die man jemals bekommt.
- Aus den Testdaten — die sind am wichtigsten.
- ✔ **Nur aus den Trainingsdaten — und wendet sie dann unverändert auf neue Daten an.**

Sonst fließt Wissen über die Prüfdaten heimlich ins Training ein („Data Leakage“) — das Modell wirkt besser, als es ist. Mehr dazu in Kapitel 235.

Das nimmst du mit

- **Tensoren dürfen beliebige Zahlen enthalten.** Die 0–1-Regel ist kein Gesetz, sondern eine Vorbereitung fürs Training.
- **Die größte Spalte dominiert alles.** 320 gegen 7: in der einen Richtung explodiert der Gradient, in der anderen kriecht er — eine Lernrate für beide gibt es dann nicht.
- **Min-Max presst jede Spalte auf 0–1.** `(x − min) / (max − min)`, spaltenweise gerechnet und per Broadcasting auf alle Zeilen angewendet.
- **Z-Score ist die robustere Variante.** `(x − Mittelwert) / Streuung` → Mittelwert 0, Streuung 1, keine feste Obergrenze — steckt Ausreißer besser weg.
- **Die Skalierungs-Werte gehören zum Modell.** Nur aus den Trainingsdaten berechnen und unverändert auf neue Daten anwenden — sonst sickert Wissen über die Prüfdaten ins Training.

So zitierst du dieses Kapitel

HerrKI: Alles auf eine Skala (Normalisierung). herrki.de/135_daten_vorbereiten, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.