Eins von vielen (Multi-Class-Klassifikation)

Kapitel online

Nicht nur ja/nein, sondern eins von vielen: mehrere Kategorien unterscheiden mit Softmax und One-Hot-Encoding.

Fortgeschritten · Kapitel 22 von 50 · ca. 25 Min. · 250 XP

Einfach gesagt: Apfel, Banane oder Orange? Dein Netz verteilt 100 Prozent Wahrscheinlichkeit auf alle Möglichkeiten, und die größte Zahl gewinnt.

Kennst du die Obst-Waagen im Supermarkt, die von selbst erkennen, was du aufgelegt hast? Oder deine Foto-App, die Bilder in „Essen“, „Tiere“, „Strand“ sortiert? Das ist die Königsdisziplin nach dem Ja/Nein aus dem letzten Kapitel: **Multi-Class-Klassifikation** — eine Entscheidung zwischen VIELEN Möglichkeiten.

Was du hier lernst

- Warum ein einzelnes Sigmoid bei 3+ Klassen nicht mehr reicht
- Wie **Softmax** die Ausgabe in eine **Wahrscheinlichkeits-Verteilung** verwandelt (Summe immer 100 %)
- Wie Kategorien selbst zu Tensoren werden: **One-Hot-Encoding**
- Der passende Fehler-Messer dazu: **Categorical Crossentropy**

Für den Unterricht

Einstieg (5 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (15 Min): die interaktiven Module auf herrki.de/230_multiclass_klassifikation ausprobieren.

Sichern mit Quiz (5 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Runter zum Tensor: Wie wird „Banane“ zu Zahlen?

Zwei Übersetzungen musst du sehen — dann ist der Rest Routine. Wir nehmen die rote, 8 cm große, 180 g schwere Frucht, die unten im Vorhersage-Formular voreingestellt ist:

1. Der Input ist wie immer ein Feature-Tensor [5]. Farbe kennst du schon als Zahlen aus „Bilder als Tensoren“ (RGB 0–255):

```
  Größe Gewicht Rot Grün Blau
[ 8 , 180 , 220 , 50 , 50 ]
```

2. Das Label — hier passiert der Aha-Moment: „Apfel“ kann das Netz nicht rechnen. Also bekommt *jede Klasse ihren eigenen Platz* im Tensor, und genau EINE Stelle ist heiß (one hot):

```
         Apfel Banane Orange
Apfel = [ 1 , 0 , 0 ]
Banane = [ 0 , 1 , 0 ]
Orange = [ 0 , 0 , 1 ]
```

(Im Modul unten kommt noch die Traube dazu — dann sind es eben vier Plätze statt drei.)

Und die Antwort des Netzes? Ein Tensor derselben Form — aber mit Wahrscheinlichkeiten. Die letzte Schicht spuckt erst drei rohe Punktzahlen aus, dann macht **Softmax** daraus eine Verteilung — hier für unseren roten Achtzentimeter:

```
Rohe Punkte: Apfel 2.50 Banane 1.25 Orange 0.55

e-Funktion: e^2.50 = 12.18
              e^1.25 = 3.49
              e^0.55 = 1.73 Summe = 17.41

Teilen: 12.18 / 17.41 = 0.70 ← 70 % Apfel
               3.49 / 17.41 = 0.20 ← 20 % Banane
               1.73 / 17.41 = 0.10 ← 10 % Orange
                              1.00 ← immer genau 100 %
```

Softmax drückt also alle Ausgaben auf 0–1 UND zwingt ihre Summe auf genau 1,0 — die Klassen konkurrieren um die 100 %. Gewonnen hat der höchste Wert. Sigmoid war die Ein-Personen-Wahl, Softmax ist die Abstimmung. Und die **Categorical Crossentropy** schaut sich davon nur die eine Zahl an der heißen Stelle des Labels `[1, 0, 0]` an:

```
Fehler = −ln(0.70) = 0.36 // 70 % auf den richtigen Apfel: billig
Fehler = −ln(0.05) = 3.00 // nur 5 % auf den Apfel, forsch daneben: 8× teurer
```

Trainingsdaten

Das Netz lernt aus **28 Beispiel-Früchten** (7 je Sorte). Wirf einen Blick auf die RGB-Spalten: Kannst du die Orangen schon an den Zahlen erkennen, bevor die Farbe daneben steht?

Nr.	Größe (cm)	⚖ Gewicht (g)	● Rot	● Grün	● Blau	Farbe	Kategorie

Training starten

> **Hyperparameter**
>
> Als Fehler-Messer arbeitet jetzt die **Categorical Crossentropy**: Sie bestraft das Netz umso härter, je selbstsicherer es auf die *falsche* Klasse tippt — ein zögerliches „33/33/33“ tut weniger weh als ein forsches „95 % Banane“, wenn es ein Apfel war.

Ausgeschrieben heißt sie **Categorical Cross-Entropy** (kategoriale Kreuzentropie) — dasselbe Maß wie die Binary Cross-Entropy aus Kapitel 220, nur mit mehr als zwei Klassen. Softmax (der Verteiler, den du auch aus „Wie LLMs Fragen beantworten“ kennst) liefert je Klasse eine Wahrscheinlichkeit, und der Fehler ist wieder **−log(Wahrscheinlichkeit der richtigen Klasse)**. Die falschen Klassen stehen gar nicht in der Formel — sie zählen nur indirekt mit, weil Softmax alle Werte auf zusammen 1,0 zwingt: Wer der falschen Frucht Wahrscheinlichkeit schenkt, nimmt sie der richtigen weg.

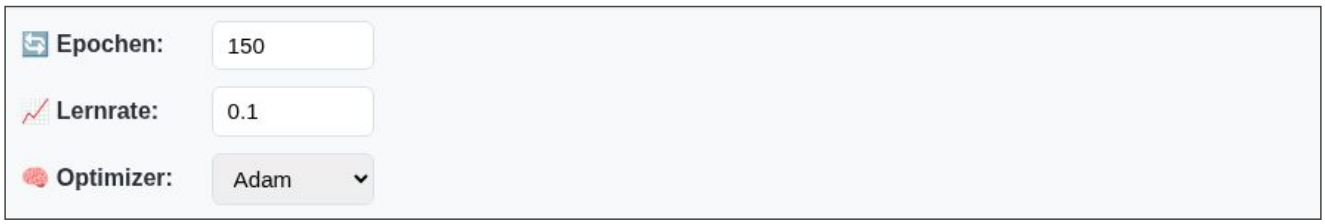
```
Richtig ist BANANE, One-Hot-Ziel [0, 1, 0]

Softmax [Apfel 0,2 | Banane 0,7 | Orange 0,1] → −ln(0,7) = 0,357 // sitzt
Softmax [Apfel 0,6 | Banane 0,3 | Orange 0,1] → −ln(0,3) = 1,204 // aufs falsche Obst gesetzt: 3,4× teurer
```

Jetzt siehst du auch, warum das One-Hot-Encoding von oben so gut dazu passt: Das Ziel `[0, 1, 0]` ist selbst eine Wahrscheinlichkeits-Verteilung („100 % Banane, sonst nichts“), und die Kreuzentropie ist genau das Maß, das zwei Verteilungen miteinander vergleicht — Ziel gegen Vorhersage. Weil im Ziel überall Nullen stehen außer an der heißen Stelle, fällt der ganze Rest der Rechnung weg, und übrig bleibt das eine −log. Im Modul unten ist das eine einzige Zeile: `loss: 'categoricalCrossentropy'`.

Am Gerät ausprobieren: herrki.de/230_multiclass_klassifikation#abschnitt-4

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

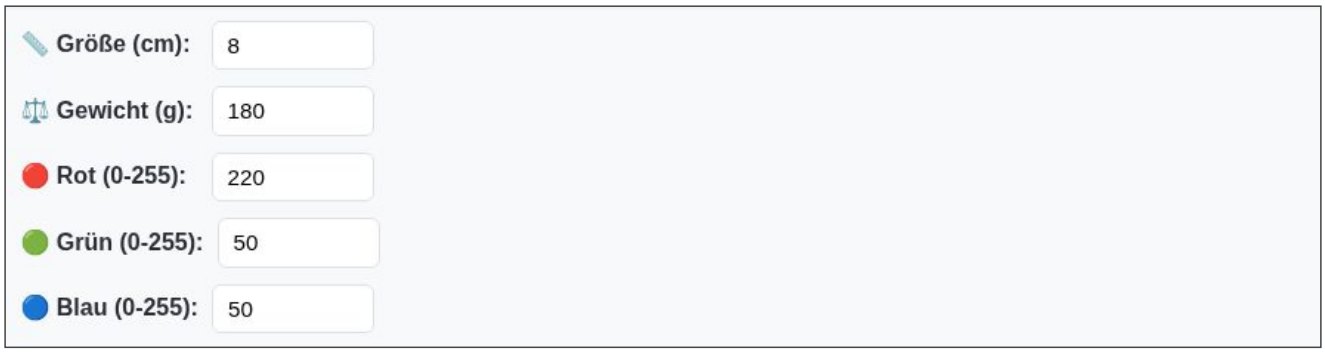
0%

Früchte klassifizieren

Bau dir dein eigenes Obst! Stell Größe, Gewicht und Farbe ein und schau, wie sich die Softmax-Verteilung entscheidet. Spannend wird's an den Grenzen: Was passiert bei einer *grünen, apfelgroßen* Frucht — und wie sicher ist sich das Netz dann noch?

Am Gerät ausprobieren: herrki.de/230_multiclass_klassifikation#abschnitt-5

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Zurück zur Anwendung

Zurück an die Obst-Waage im Supermarkt: Du legst deine Frucht auf, und statt einer Tastenwand erscheint „Apfel — 96 %“. Was da passiert, kennst du jetzt vollständig — Merkmale werden zu einem Tensor, das Netz spuckt rohe Punktzahlen aus, Softmax verteilt daraus 100 % auf alle Sorten, und die größte Zahl gewinnt. Genauso sortiert deine Foto-App in „Essen“, „Tiere“ und „Strand“, und genauso entscheidet eine Sprach-Erkennung zwischen „an“, „aus“ und „lauter“. Das Schöne daran: Weil immer eine ganze Verteilung herauskommt, kannst du auch sehen, *wie knapp* es war — 70 % Apfel und 20 % Banane ist eine ganz andere Aussage als 99 % Apfel.

Bleibt eine unbequeme Frage — die Früchte hier kannte das Netz ja schon. Wie man ehrlich prüft, ob ein Modell auch mit Unbekanntem klarkommt, steht im nächsten Kapitel: „Trainieren, prüfen, bestehen“.

Zum Weiterlesen

- **Mehrklassen-Netze kompakt erklärt:** Google Machine Learning Crash Course, *Multi-class classification* — developers.google.com → Multi-class classification (developers.google.com/machine-learning/crash-course/neural-networks/multi-class)
- **Softmax und Cross-Entropy im Lehrbuch:** Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, Kapitel 6 (frei online) — deeplearningbook.org/contents/mlp.html
- **Softmax hergeleitet:** Christopher M. Bishop, *Pattern Recognition and Machine Learning*, 2006, Kapitel 4 — PDF gratis bei Microsoft Research (18 MB) (microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf) — dort steht, warum die Softmax-Funktion die Verallgemeinerung der logistischen Regression auf mehrere Klassen ist

Kurzer Check – mit Lösung

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

Frage 1: Deine Klassen sind [Apfel, Banane, Orange]. Wie sieht das One-Hot-Label für „Banane" aus?

- ✔ **[0, 1, 0]**
- [1, 1, 0]
- [2]

Jede Klasse bekommt ihren festen Platz, genau eine Stelle ist „heiß". So kann das Netz sein Wahrscheinlichkeits-Ergebnis direkt mit dem Ziel-Tensor vergleichen.

Frage 2: Softmax gibt [0,70 · 0,20 · 0,10] aus. Was gilt bei Softmax IMMER?

- Der größte Wert liegt immer über 0,9.
- ✔ **Die Werte summieren sich zu 1 — es ist eine Wahrscheinlichkeits-Verteilung über alle Klassen.**
- Alle Klassen bekommen den gleichen Wert.

Softmax ist die „Abstimmung um 100 %": Die Klassen teilen sich die gesamte Wahrscheinlichkeit. Hier: ziemlich sicher Apfel, aber mit ehrlicher Rest-Unsicherheit.

Frage 3: Warum „bestraft" die Crossentropy ein Modell besonders hart, das selbstsicher falsch liegt?

- ✔ **Damit das Netz lernt, bei Unsicherheit auch unsichere Wahrscheinlichkeiten auszugeben statt zu bluffen.**
- Weil falsche Antworten mehr Speicher kosten.
- Das stimmt nicht — Crossentropy behandelt alle Fehler gleich.

„Selbstsicher falsch tut mehr weh": 95 % auf die falsche Frucht gibt eine viel höhere Strafe als vorsichtige 40 %. Genau das erzieht das Netz zu ehrlichen Wahrscheinlichkeiten.

Das nimmst du mit

- **One-Hot macht Kategorien rechenbar.** Jede Klasse bekommt ihren festen Platz, genau eine Stelle ist heiß: Banane = [0, 1, 0].
- **Softmax verteilt 100 %.** e-Funktion, teilen, fertig — die Ausgaben summieren sich immer auf genau 1,00.
- **Sigmoid war die Ein-Personen-Wahl, Softmax ist die Abstimmung.** Die Klassen konkurrieren um dieselbe Wahrscheinlichkeit; die höchste gewinnt.
- **Categorical Crossentropy schaut nur auf die heiße Stelle.** $-\ln(0{,}70) = 0{,}36$ gegen $-\ln(0{,}05) = 3{,}00$ — achtmal teurer.
- **Selbstsicher falsch tut am meisten weh.** Genau das erzieht das Netz zu ehrlichen Wahrscheinlichkeiten statt zum Bluffen.

Bevor du weiterziehst, eine unbequeme Frage: Dein Netz trifft hier vielleicht 100 % — auf Früchten, *die es schon kennt*. Aber würde es auch die Orange vom Wochenmarkt erkennen, die es nie gesehen hat? Wie man das ehrlich prüft, ist Thema des nächsten Kapitels: „Trainieren, prüfen, bestehen“.

So zitierst du dieses Kapitel

HerrKI: Eins von vielen (Multi-Class-Klassifikation). herrki.de/230_multiclass_klassifikation, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.