Lern-Nugget: Ein einzelnes Neuron verstehen

🎯 Ziel

Ein einzelnes Neuron als zusammengesetzte Funktion f(x) = g(h(x)) verstehen - die Grundbausteine neuronaler Netze!

🧠 Was ist ein Neuron?

Ein Neuron = Lineare Transformation + Nichtlineare Aktivierung

Das ist genau die Struktur einer zusammengesetzten Funktion:

- h(x) = Lineare Funktion (gewichtete Summe + Bias)
- g(x) = Aktivierungsfunktion (Nichtlinearität)
- f(x) = g(h(x)) = Das komplette Neuron

📐 Konkrete Funktionen definieren

Einfaches Beispiel: Ein Input

Die lineare Komponente h(x):

- h(x) = wx + b = 2x + 1
- (Gewicht w = 2, Bias b = 1)

Die Aktivierungsfunktion g(h):

- g(h) = 1/(1 + e^(-h)) = Sigmoid-Aktivierung
- (Macht aus beliebigen Zahlen Werte zwischen 0 und 1)

Das komplette Neuron: f(x) = g(h(x)) = sigmoid(2x + 1)

Realistisches Beispiel: Zwei Inputs

Die lineare Komponente $h(x_1, x_2)$:

- $h(x_1, x_2) = w_1x_1 + w_2x_2 + b = 2x_1 + 3x_2 + 1$
- (Gewichte $w_1 = 2$, $w_2 = 3$, Bias $b = 1$)

Die Aktivierungsfunktion g(h):

- g(h) = sigmoid(h) = 1/(1 + e^(-h))

Das komplette Neuron: $y = g(h(x_1, x_2)) = \text{sigmoid}(2x_1 + 3x_2 + 1)$

🔍 Was macht jede Komponente?

Schritt 1: Lineare Transformation h(x) = 2x + 1

Beispielwerte:

- x = 0 → h = 2·0 + 1 = 1
- x = 1 → h = 2·1 + 1 = 3
- x = -1 → h = 2·(-1) + 1 = -1

Was passiert: Input wird skaliert (×2) und verschoben (+1)

Schritt 2: Sigmoid-Aktivierung g(h) = 1/(1 + e^(-h))

Beispielwerte:

- h = 1 → g = 1/(1 + e^(-1)) ≈ 0.73
- h = 3 → g = 1/(1 + e^(-3)) ≈ 0.95
- h = -1 → g = 1/(1 + e^(1)) ≈ 0.27

Was passiert: Beliebige Zahlen werden auf (0,1) "gequetscht"

Das komplette Neuron f(x):

- x = 0 → f(0) = sigmoid(1) ≈ 0.73
- x = 1 → f(1) = sigmoid(3) ≈ 0.95
- x = -1 → f(-1) = sigmoid(-1) ≈ 0.27

⛓️ Die Kettenregel für das Neuron

Was wir berechnen wollen: f'(x)

Die einzelnen Ableitungen:

Ableitung der linearen Komponente:

- h(x) = 2x + 1
- dh/dx = 2

Ableitung der Sigmoid-Funktion - Schritt für Schritt:

Gegeben: g(h) = 1/(1 + e^(-h)) = (1 + e^(-h))^(-1)

Schritt 1: Kettenregel anwenden

- Äußere Funktion: u^(-1), Ableitung: -u^(-2)
- Innere Funktion: u = 1 + e^(-h), Ableitung: -e^(-h)

Schritt 2: Zusammensetzen

- dg/dh = -1 · (1 + e^(-h))^(-2) · (-e^(-h))
- dg/dh = e^(-h) / (1 + e^(-h))²

Schritt 3: Elegant umformen

- dg/dh = e^(-h) / (1 + e^(-h))²
- = e^(-h) / (1 + e^(-h))² · (1 + e^(-h)) / (1 + e^(-h))
- = e^(-h) / (1 + e^(-h)) · 1 / (1 + e^(-h))
- = [1 - 1/(1 + e^(-h))] · 1/(1 + e^(-h))

- **= [1 - g(h)] · g(h) = g(h) · (1 - g(h))**

🎯 **Elegant:** sigmoid'(h) = sigmoid(h) · (1 - sigmoid(h))

Warum ist diese Form so praktisch?

- Wir müssen keine komplizierten Exponentialfunktionen berechnen!
- Wir brauchen nur den **bereits berechneten** Sigmoid-Wert
- Forward Pass: y = sigmoid(h) berechnen
- Backward Pass: y' = y · (1 - y) verwenden ← **Superschnell!**

Kettenregel anwenden:

f'(x) = (dg/dh) · (dh/dx)

Konkret für x = 0:

- h(0) = 1
- g(1) ≈ 0.73
- dg/dh = 0.73 · (1 - 0.73) = 0.73 · 0.27 ≈ 0.20
- dh/dx = 2
- **f'(0) = 0.20 · 2 = 0.40**

🔍 Der Sprung zu realistischen Neuronen: Zwei Inputs

Jetzt wird es realistisch! Neuron mit zwei Inputs:

- y = sigmoid($2x_1 + 3x_2 + 1$)

Warum partielle Ableitungen (*∂*)?

Das Neuron hängt von mehreren Variablen ab:

- y = y(x_1, x_2) ← Funktion von **zwei** Variablen!

Partielle Ableitungen berechnen:

$\partial y/\partial x_1$ (wie ändert sich y, wenn nur x_1 sich ändert?)

- h = $2x_1 + 3x_2 + 1$
- $\partial h/\partial x_1 = 2$
- $\partial y/\partial h$ = sigmoid'(h)
- **$\partial y/\partial x_1$ = sigmoid'(h) · 2**

$\partial y/\partial x_2$ (wie ändert sich y, wenn nur x_2 sich ändert?)

- $\partial h/\partial x_2 = 3$
- **$\partial y/\partial x_2$ = sigmoid'(h) · 3**

🎯 Kettenregel mit partiellen Ableitungen:

$\partial y/\partial x_1 = (\partial y/\partial h) \cdot (\partial h/\partial x_1)$ $\partial y/\partial x_2 = (\partial y/\partial h) \cdot (\partial h/\partial x_2)$

Konkret für $x_1 = 1$, $x_2 = 0.5$:

- h = 2·1 + 3·0.5 + 1 = 4.5
- y = sigmoid(4.5) ≈ 0.989
- ∂y/∂h = 0.989 · (1 - 0.989) ≈ 0.011
- **$\partial y/\partial x_1 = 0.011 \cdot 2 = 0.022$**
- **$\partial y/\partial x_2 = 0.011 \cdot 3 = 0.033$**

Intuitive Erklärung

Was sagt uns die Ableitung?

f'(x) = sigmoid'(h) · w

- **sigmoid'(h):** "Wie stark reagiert die Aktivierungsfunktion?"
- **w:** "Wie stark ist die lineare Transformation?"
- **Produkt:** "Wie stark reagiert das gesamte Neuron auf Input-Änderungen?"

Interessante Beobachtungen:

- Bei h ≈ 0: sigmoid'(h) ist maximal → Neuron reagiert stark
- Bei h → ±∞: sigmoid'(h) → 0 → Neuron "sättigt", reagiert schwach
- Größeres |w|: Neuron reagiert stärker auf Input-Änderungen

Warum ist das für neuronale Netze wichtig?

Beim Training brauchen wir:

1. **Forward Pass:** f(x) = sigmoid(wx + b) berechnen
2. **Backward Pass:** f'(x) = sigmoid'(wx + b) · w berechnen

Die Kettenregel gibt uns beides:

- Die Aktivierung des Neurons
- Den Gradienten für Backpropagation

In größeren Netzen:

```
Input → Neuron₁ → Neuron₂ → ... → Output → Loss
```

Jedes Neuron ist ein f(x) = g(h(x)), und die Kettenregel verbindet sie alle!

Zusammenfassung

Ein Neuron ist eine zusammengesetzte Funktion:

- **h(x) = wx + b** (Lineare Transformation)
- **g(h) = sigmoid(h)** (Nichtlineare Aktivierung)
- **f(x) = g(h(x))** (Komplettes Neuron)

Die Kettenregel gibt uns:

- **Forward:** f(x) berechnen
- **Backward:** f'(x) = g'(h(x)) · h'(x) berechnen

Kernidee: Jedes Neuron kombiniert eine einfache lineare Transformation mit einer nichtlinearen Aktivierung. Die Kettenregel ermöglicht es, durch diese Kombination zu differenzieren!

🎯 Übung: Partielle Ableitungen

Probiere selbst - ReLU-Neuron mit zwei Inputs:

- $h(x_1, x_2) = 2x_1 + x_2 - 1$
- $g(h) = \max(0, h)$ (ReLU-Aktivierung)
- $y = g(h(x_1, x_2)) = \max(0, 2x_1 + x_2 - 1)$

Berechne die partiellen Ableitungen:

- $\partial h/\partial x_1$ = ?
- $\partial h/\partial x_2$ = ?
- $\partial g/\partial h$ = ? (ReLU-Ableitung ist 1 wenn h > 0, sonst 0)
- $\partial y/\partial x_1$ = ?
- $\partial y/\partial x_2$ = ?

Lösung:

- $\partial h/\partial x_1 = 2$, $\partial h/\partial x_2 = 1$
- $\partial g/\partial h = 1$ wenn h > 0, sonst 0
- $\mathbf{\partial y/\partial x_1 = 2}$ wenn $2x_1 + x_2 > 1$, sonst 0
- $\mathbf{\partial y/\partial x_2 = 1}$ wenn $2x_1 + x_2 > 1$, sonst 0

🎯 **Wichtig:** Bei mehreren Inputs brauchen wir **partielle Ableitungen** (∂)!

💡 **Tipp:** Verstehe ein einzelnes Neuron, und du verstehst die Grundlage aller neuronalen Netze. Jedes große Netz ist nur eine Verkettung vieler solcher f(x) = g(h(x)) Funktionen!