KI für Alle 2: Verstehen, Bewerten, Reflektieren

Themenblock Prognosemodelle (Klassifikation und Regression): 02_03Verfahren_NeuronaleNetze_02

# Neuronale Netze – Teil 2

Erarbeitet von
Dr. Katja Theune



## Lernziele

- Du kannst den Verlust anhand eines einfachen Beispiels einordnen
- Du kannst den Prozess, wie ein neuronales Netz lernt (insbesondere die backpropagation), erläutern
- Du kannst die Problematik bei der Wahl der Hyperparameter erläutern
- Du kannst Vor- und Nachteile neuronaler Netze erläutern



Projekt ai4all „KI für Alle"
HeiCAD –Heine Center for Artificial Intelligence and Data Science

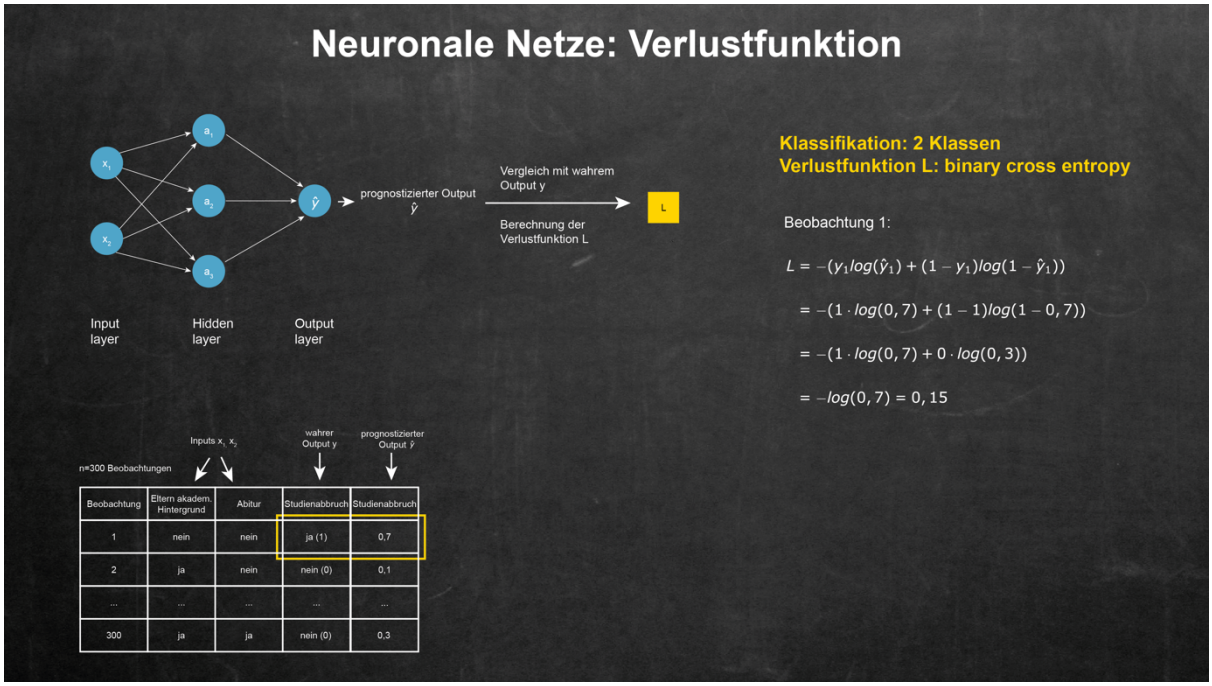
# Inhalt

## Einstieg

Künstliche neuronale Netze sind sehr flexibel und leistungsstark. Doch auch sie machen Fehler bei der Prognose. Diese sollen natürlich möglichst gering sein. Dafür kann unser Netz aus den Daten selbständig und schrittweise lernen, wie es möglichst passende Vorhersagen macht. Doch wie genau sieht dieses Lernen aus?

## Verlustfunktion: Idee und Beispiel

Um beurteilen zu können, ob eine Prognose gut war oder nicht, müssen wir den vom Netz prognostizierten Output mit dem wahren Output vergleichen und den sogenannten Fehler des Modells bestimmen. Man nennt ihn auch Verlust, oder englisch loss.
Um ihn zu berechnen, verwenden wir eine Verlustfunktion, oder auch loss function. Häufig wird sie auch Kostenfunktion bzw. cost function genannt.



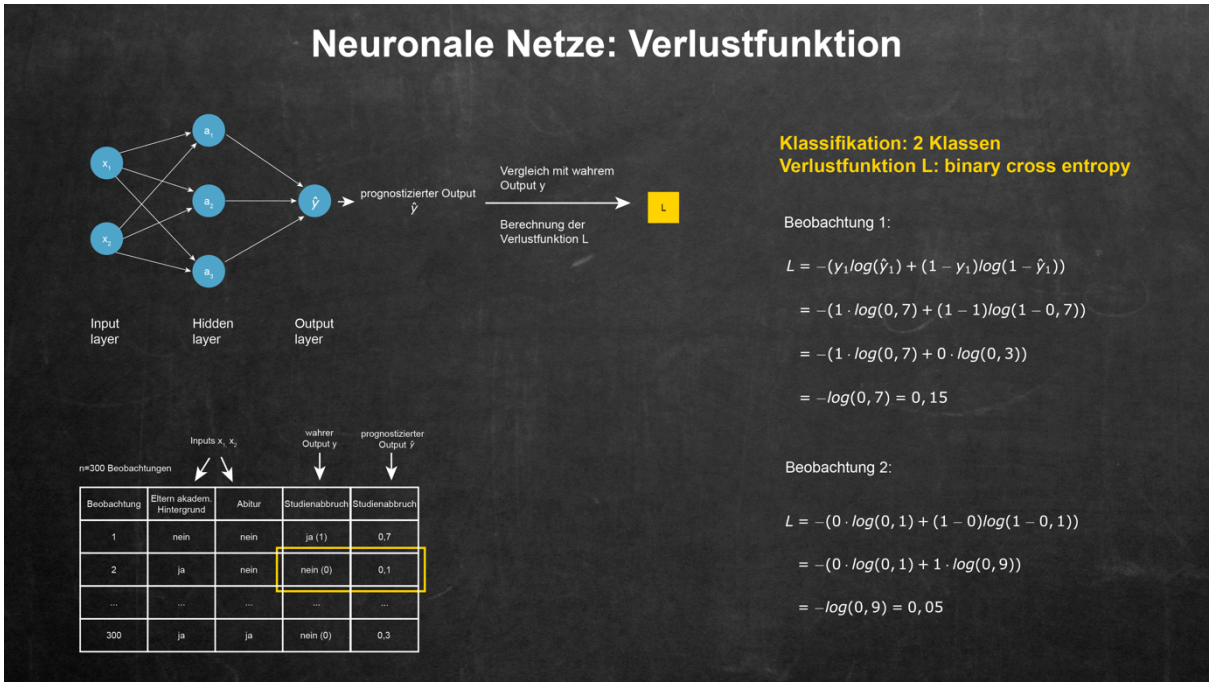

Zur Veranschaulichung wollen wir wieder mit den zwei Inputs „Akademischer Hintergrund der Eltern" und „Abitur" einen möglichen Studienabbruch prognostizieren. Wir kodieren beim wahren Output y die Klasse „Studienabbruch" mit 1 und „kein Studienabbruch" mit 0, was auch jeweils als die Wahrscheinlichkeit für einen Studienabbruch interpretiert werden kann. Als prognostizierten Output $\hat{y}$ erhalten wir durch die Sigmoid-Aktivierungsfunktion die Wahrscheinlichkeit, zu der Klasse „Studienabbruch" zu gehören.

Im Falle einer Klassifikation mit zwei Klassen wird als Verlustfunktion L häufig die binary cross entropy verwendet.

Schauen wir uns direkt als Beispiel Beobachtung 1 an. Der Verlust berechnet sich für diese Beobachtung folgendermaßen (siehe Grafik).

$y_1$ ist der wahre Wert des Outputs von Beobachtung 1. Hier ist er gleich 1, d. h. es fand ein Studienabbruch statt. $\hat{y}_1$ ist die vom neuronalen Netz prognostizierte Wahrscheinlichkeit zur Klasse Studienabbruch zu gehören, hier 0,7. Log steht für den Logarithmus. Genaueres müssen wir dazu erstmal nicht wissen. Wir erhalten also einen Verlust von 0,15 (siehe Grafik).

Schauen wir uns noch die zweite Beobachtung an. Hier haben wir einen wahren 0utput von 0, es fand demnach kein Studienabbruch statt. Die prognostizierte Wahrscheinlichkeit $\hat{y}$ beträgt hier 0,1. Wir erhalten also einen Verlust von 0,05 (siehe Grafik).



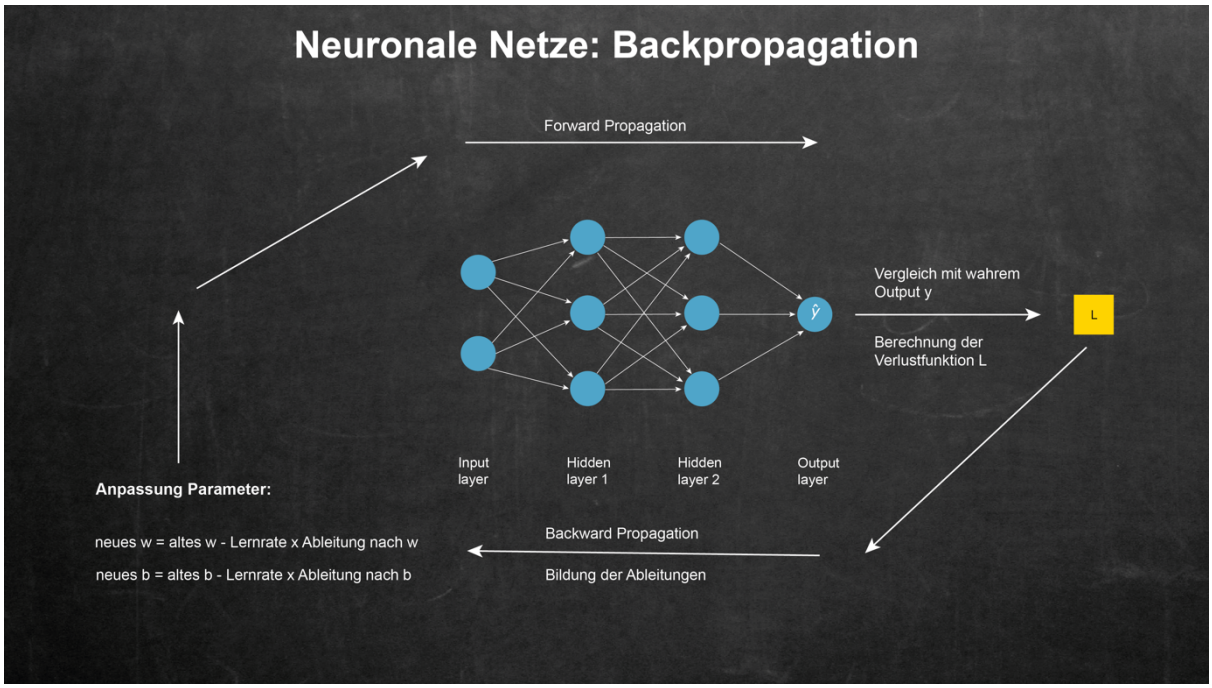

Was wir hier sehen ist, dass wir bei Beobachtung 2 mit unserer Prognose viel näher an dem wahren Wert lagen und damit auch der Verlust kleiner ausfällt als bei Beobachtung 1. Es gilt also: je besser die Prognose, desto kleiner der Verlust.

Der gesamte Verlust für alle Beobachtungen berechnet sich dann als Mittel über die Verluste aller Beobachtungen. Ganz ähnlich kann man mit der cross entropy den Verlust für einen Fall mit mehreren Klassen berechnen.

Das neuronale Netz lernt jetzt aus diesem Verlust und passt die Gewichte und Biases so an, dass es eine bessere Prognose macht und wir einen kleineren Verlust erhalten. Aber wie funktioniert das genau?

## Lernen durch Gradientenabstieg und Backpropagation

Da wir für den ersten Durchlauf zunächst meist rein zufällige oder von uns gewählte Initialwerte für die Gewichte w und Biases b verwenden, sind falsche Prognosen und hohe Werte der Verlustfunktion zu Beginn des Trainings natürlich nicht ungewöhnlich. Eine übliche Methode, um diese beiden Parameter nach und nach zu optimieren und den Verlust zu minimieren, ist das Verfahren des Gradientenabstiegs, bzw. gradient descent, in Verbindung mit der backward propagation, bzw. backpropagation.




Das Gradientenabstiegsverfahren soll hier jetzt kein Thema sein, nur kurz dazu: es ist ein Optimierungsverfahren, mit dessen Hilfe wir das Minimum der Verlustfunktion finden können. Die Minimierung des Verlustes erreichen wir hier, indem wir die Gewichte und Biases schrittweise in die entgegengesetzte Richtung des Gradienten der Verlustfunktion bzgl. dieser Parameter anpassen.

Bei sehr großen Netzen müssen wir auch sehr viele optimale Gewichte und Biases finden und daher sehr komplexe Gradienten mit sehr vielen Ableitungen berechnen. Dafür eignet sich die backpropagation. Die genauen Berechnungen schauen wir uns nicht weiter an, aber immerhin kurz die Idee.

Bisher haben wir uns von vorne nach hinten durch das neuronale Netz durchgearbeitet und in jedem Layer gewichtete Summen gebildet und darauf Aktivierungsfunktionen angewendet, um am Ende unseren Output bzw. unsere Prognose zu erhalten. Wir haben sozusagen eine forward propagation vollzogen.

Jetzt drehen wir das ganze um und arbeiten uns, wie der Name backpropagation schon suggeriert, von hinten nach vorne durchs neuronale Netz. Der Fehler, den jede einzelne Schicht mit seinen Gewichten und Biases zu dem abschließenden Prognosefehler beiträgt,

hängt auch von den darauffolgenden Schichten ab, denn diese geben den Fehler in einer bestimmten Art und Weise weiter. Daher ist es sinnvoll, mit der Fehlerminimierung ganz hinten zu starten und sich dann Schicht für Schicht zurückzuarbeiten. Der Fehler wird also von hinten nach vorne zurückgeführt und so die verketteten Ableitungen gebildet.
Die Anpassung der Gewichte und Biases erfolgt, indem wir von den ursprünglichen Werten die jeweilige Ableitung multipliziert mit der Lernrate abziehen. Das ist sinnvoll, da wir ja den Verlust minimieren wollen. Zur Erinnerung: Die Lernrate gibt die Schrittgröße an, mit denen wir uns dem Minimum nähern.

Nach der Anpassung der Parameter können wir wieder mit der forward propagation starten. Diesmal aber mit den nun aktualisierten Parametern. Dieses Vorgehen der back- und forward propagation wiederholen wir mehrfach und nähern uns dem Minimum immer weiter an, indem wir, bildlich gesprochen, den Berg der Verlustfunktion hinabsteigen. Man nennt die verschiedenen Durchläufe auch Epochen.

## Die Wahl von Hyperparametern

Zu den Hyperparametern bei den neuronalen Netzen gehören z. B. die Lernrate, die Anzahl an Layern und Neuronen oder auch die Anzahl an Epochen. Und auch hier begegnen wir häufig einem Trade-off zwischen Verzerrung und Varianz bei der Auswahl dieser Parameter. So ist es bei komplexen Problemen und Daten sinnvoll, viele Layer und Neuronen zu verwenden. Große Netze können diese komplexen Zusammenhänge dann sehr gut lernen und liefern gute Prognosen mit einer kleinen Verzerrung. Allerdings besteht hier dann die Gefahr einer hohen Varianz und Overfitting, insbesondere bei kleineren Datenmengen. Die Ergebnisse lassen sich dann nicht gut auf andere Daten übertragen. Wählen wir zu wenige Layer und Neuronen, dann werden die komplexen Muster ggf. nicht gut genug gelernt und wir erhalten eine hohe Verzerrung, aber eine kleinere Varianz.

## Diskussion, Vor- und Nachteile

Ein großer Vorteil ist, dass neuronale Netze sehr flexibel sind. Durch das Verfahren des Gradientenabstiegs und der backward propagation kann das neuronale Netz Schritt für Schritt aus den Daten selbstständig lernen, seine Parameter, also die Gewichte und Biases, aktualisieren und so komplexe Strukturen und Muster in den Daten erkennen. Wir erhalten dann sehr gute Prognosen.

Bei großen Netzen müssen aber sehr viele Parameter gelernt werden, was sehr viele Daten benötigt. Zudem ist es auch sehr rechen- und zeitaufwendig, diese Parameter zu lernen.

## Abschluss

Wir wissen nun, wie ein neuronales Netz selbstständig aus den Daten lernt, indem es den Fehler bei seiner Prognose berechnet und ihn sozusagen als Feedback für die eigene Verbesserung verwendet. Das Gradientenabstiegsverfahren und die backpropagation sind dafür übliche Methoden, die zwar rechenaufwendig sind, aber zu sehr guten Prognosen des neuronalen Netzes führen.

## Weiterführendes Material

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3. Auflage). Morgan Kaufmann.

James, G., Witten, D., Hastie, T., & Tibshirani, R., & Tylor, J. (2023). *An Introduction to Statistical Learning - with Applications in Python*. Springer.

Lantz, B. (2015). *Machine learning with R* (2. Auflage). Packt Publishing Ltd, Birmingham.

Videoreihe zu neuronalen Netzen:

3Blue1Brown: Neural networks

https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

## Disclaimer